SCHEDULING AND RESOURCE EFFICIENCY BALANCING

Discrete Species Conserving Cuckoo Search for Scheduling in an Uncertain Execution Environment

Kirils Bibiks

Submitted for the Degree of
Doctor of Philosophy

Faculty of Engineering and Informatics
University of Bradford

2017

Abstract

Kirils Bibiks

Scheduling and Resource Efficiency Balancing

Discrete Species Conserving Cuckoo Search for Scheduling in an Uncertain Execution Environment

Keywords: Project Scheduling, Cuckoo Search, Species Conservation, Combinatorial Optimisation, Evolutionary Computation.

The main goal of a scheduling process is to decide when and how to execute each of the project's activities. Despite large variety of researched scheduling problems, the majority of them can be described as generalisations of the resource-constrained project scheduling problem (RCPSP). Because of wide applicability and challenging difficulty, RCPSP has attracted vast amount of attention in the research community and great variety of heuristics have been adapted for solving it. Even though these heuristics are structurally different and operate according to diverse principles, they are designed to obtain only one solution at a time. In the recent researches on RCPSPs, it was proven that these kind of problems have complex multimodal fitness landscapes, which are characterised by a wide solution search spaces and presence of multiple local and global optima.

The main goal of this thesis is twofold. Firstly, it presents a variation of the RCPSP that considers optimisation of projects in an uncertain environment where resources are modelled to adapt to their environment and, as the result of this, improve their efficiency. Secondly, modification of a novel evolutionary computation method Cuckoo Search (CS) is proposed, which has been adapted for solving combinatorial optimisation problems and modified to obtain multiple solutions. To test the proposed methodology, two sets of experiments are carried out. First, the developed algorithm is applied to a real-life software development project. Second, performance of the algorithm is tested on universal benchmark instances for scheduling problems which were modified to take into account specifics of the proposed optimisation model. The results of both experiments demonstrate that the proposed methodology achieves competitive level of performance and is capable of finding multiple global solutions, as well as prove its applicability in real-life projects.

Acknowledgements

I would like to convey my heartfelt gratitude and sincere appreciation to my supervisors, Professor Yim Fun Hu and Doctor Jian-Ping Li from the Faculty of Engineering and Informatics, University of Bradford, for their continuous support of my PhD study and related research, for their patience, motivation, inspiration, and immense knowledge. Their guidance helped me at all times during the work and research on this thesis. I could not have imagined having better advisors and mentors for my study.

I thank my fellow labmates with whom we have worked on many projects and assignments for all that time we spent together completing our work before the deadlines and for all the fun we have had in the last three years.

There is a long list of friends who have been very supportive and encouraging and without whom I might not have reached the end of this study. I am thankful to all of them.

Finally, this thesis is dedicated to my parents, family members and my teachers for all their support, encouragement, and good wishes.

Table of Contents

List of Figures

List of Tables

Acronyms

ACO	Ant Colony Optimisation
AL	Activity List
AON	Activity-On-Node
ASL	Activity Set List
CP	Critical Path
CS	Cuckoo Search
DSCCS	Discrete Species Conserving Cuckoo Search
DCS	Discrete Cuckoo Search
DE	Differential Evolution
DFPA	Discrete Flower Pollination Algorithms
DP	Dynamic Programming
EC	Evolutionary Computation
FS	Fitness Sharing
FPA	Flower Pollination Algorithm
GA	Genetic Algorithm
GLS	Guided Local Search
GPP	Graph Planarization Problem
GRASP	Greedy Randomised Adaptive Search Procedure
HARNet	Harmonised Antennas and Radio Networks
HATS	HARNet Automated Testing System
IDCS	Improved Discrete Cuckoo Search
ILS	Iterated Local Search
JSSP	Job Shop Scheduling Problem
LOP	Linear Ordering Problem
MAXSAT	Maximum Satisfaction
MMGA	Multi-Modal Genetic Algorithm
MRCPSP	Multi-mode Resource-Constrained Project Scheduling Problem
NMA	Niching Memetic Algorithm
PERT	Project Evaluation Review Technique
PRBM	Priority Rule-Based Method

PSO	Particle Swarm Optimisation
PSPLIB	Project Scheduling Problems Library
QAP	Quadratic Assignment Problem
RCMPSP	Resource-Constrained Multi-Project Scheduling Problem
RCPSP	Resource-Constrained Project Scheduling Problem
RK	Random Key
RTS	Restricted Tournament Selection
SA	Simulated Annealing
SCGA	Species Conservation Genetic Algorithm
SGS	Schedule Generations Scheme
SOA	Service Oriented Architecture
SOP	Sequential Ordering Problem
SRCPSP	Stochastic Resource-Constrained Project Scheduling Problem
SC	Species Conservation
SS	Scatter Search
TS	Tabu Search
TSP	Travelling Salesman Problem
VNS	Variable Neighbourhood Search
VRP	Vehicle Routing Problem
WP	Work Package

Glossary

Activity – Smallest unit of work that has the following characteristics: definite duration, logic relationship with other activities, and resource requirements.

Combinatorial optimisation – The process of searching for maxima (or minima) of an objective function whose domain is a discrete but large configuration space.

Continuous optimisation – The process of finding the minimum or maximum value of a function of one or many real variables, subject to constraints that take form of inequalities.

Critical path – The duration of the longest activity sequence of a project obtained by relaxing resource constraints of the problem.

Global optimum – A solution that is optimal among all possible solutions, not just those in a particular neighbourhood of values.

Heuristic – An approach to problem solving that employs a practical method or previous experience with similar problem. Heuristic is not guaranteed to obtain optimal solution, but sufficient for immediate goals.

Local optimum – The best solution to a problem within a small neighbourhood of possible solutions.

Makespan – Total duration of a project.

Metaheuristic – A high-level problem-independent algorithmic framework that provides a set of guidelines or strategies needed to develop an algorithm for an optimisation problem.

Multimodal optimisation – The process of finding multiple optimal and/or local solutions with the aim of complex optimisation problems.

Project – Set of interrelated activities that are to be executed over a fixed period of time and set of resources that are to be used for the execution of activities.

Project network – A set of nodes and arcs that depict project activities and precedence relations and model technological relations between pairs of activities.

Project scheduling problem library – Library that contains different problem sets for various types of resource-constrained project scheduling problems.

Resource-constrained project scheduling problem – Combinatorial optimisation problem objective of which is to find a feasible schedule of minimal duration, obtained by assigning a start time to each activity such that the precedence relations and the resource availabilities are respected.

Resources – People, equipment, facilities, funding, or anything else capable of definition required for the completion of a project activity.

Schedule – Timetable for a project that shows how activities are sequenced and when they are going to be executed.

Schedule generation scheme – An algorithm that transforms a solution representation scheme into a schedule.

Solution representation scheme – Scheme that determines how the problem is structured in the applied algorithm and influences the applied operators.

Solution search space – The space of all feasible solutions or the set of solutions among which the desired solution resides..

Chapter 1 Introduction

Nowadays, planning and management of resources is an increasingly important issue not just in engineering, but in all spheres of business in general. Thus, a careful management of projects, whether it is a software, construction, budgetary or any other type of project, is an absolute necessity to preserve efficient and stable operation. The biggest role in project management is devoted to scheduling. The main goal of the scheduling process is to decide when to start each of the project's activities and how these activities will share the available resources. When such decisions are made, they are expected to have a large impact on the total duration of the project (i.e. the makespan) and the overall efficiency of resource use. Consequently, the outcomes of these decisions (makespan and resource efficiency balance) are considered to be the main performance criteria when assessing the quality of the optimised project plan. Nevertheless, when planning such decisions there is a great number of challenges to be faced:

1. *Resource allocation*: a project consists of a set of activities which for their execution require different types of resources with limited capacities and different levels of efficiencies;
2. *Time dependency and presence of uncertainties*: there are many factors that affect the execution time of activities;
3. *Conflicting objectives*: often optimisation model of a project consists of multiple objectives, where optimisation of one might negatively impact optimisation of the other.

A great variety of scheduling problems has been addressed in the literature (Weglarz, 1999; Kolisch & Padman, 2001), however, despite the assortment, the majority of these problems can be modelled as generalisations of the resource-constrained project scheduling problem (RCPSP). The standard RCPSP represents a generalised version of the job-shop scheduling problem (JSSP) (Graham, 1966) and in terms of its decision variables, constraints and objective functions can be defined as follows. A set of activities and a set of resources of known characteristics (activity durations, activity resource demands, activity relations and resource availabilities) are given. The decision variables are the activity starting times, whereas the objective function is the minimisation of

project's makespan, i.e. the completion time of the last activity, assuming project starts at time 0. Two types of constraints are considered: (1) *Precedence constraints* define relationships between activities and their respective order of execution. (2) *Resource constraints* ensure that at each time period and for each resource the total activity demand does not exceed the resource availability. Once started, an activity cannot be interrupted. Other variations of the RCPSP exist as well. One of the most common of them are stochastic RCPSP (SRCPSP), in which activity durations follow some pre-defined probability distribution, and multi-mode RCPSP (MRCPSP), in which a trade-off between activity durations and resource requirements is assumed.

Nevertheless, despite simplicity of definition, RCPSPs belong to a class of NP-hard (Leeuwen, 1998) combinatorial optimisation problems (Blazewicz, Lenstra, & Kan, 1983), therefore can be considered as intrinsically harder than those that can be solved by a nondeterministic Turing machine in polynomial time. For example, in the publicly available project scheduling problems library (PSPLIB) (Kolisch & Sprecher, 1997), which contains benchmark instances for assessing the performance of algorithms for RCPSPs, the optimal makespan of instances with 60 or more activities is still unknown.

Moreover, similarly to other combinatorial optimisation problems, such as JSSP and travelling salesman problem (TSP) (Ikeda & Kobayashi, 2000), RCPSPs are proven to have complex multimodal fitness landscapes (Czogalla & Fink, 2009), which contain high amount of global optima that are spread across whole solution search space. It was demonstrated that solution search space of such problems has a big valley structure where good solutions tend to be close to other good solutions (but not too close) and are spread all around the solution search space. The statistical analysis indicated that landscapes of these problems typically consist of several interior plateau meaning that one instance of a problem can have multiple optimal solutions.

In the context of optimisation, finding a set of global solutions can be highly desirable for several reasons. First, it will help to eliminate premature convergence to local optima by diverting the search process into various regions of the search space simultaneously. Secondly, a set of diverse high-quality solutions can provide an alternative, potentially better and more innovative, outcome result in the decision making process.

For its relevance to all fields of engineering and challenging difficulty, solving the RCPSP has become a flourishing theme for a research community. This becomes even clearer when observing the amount of books (Neumann, Schwindt, & Zimmermann, 2003; Dorndorf, 2002; Klein, 2001) and research surveys (Kolisch, Sprecher, & Drexl, 1995; Lawrence, 1984; Herroelen W. , 2006; Hartmann & Briskorn, 2010) that were published on this subject.

1.1 Motivation and Objectives

Exploitation and exploration of the multimodal fitness landscapes of RCPSPs have received little research attention. Despite the variety of published research articles and surveys on RCPSPs, the majority of them aimed at optimising only one objective (i.e. makespan minimisation) and at obtaining only one solution. At the time of writing of this thesis and to the knowledge of the author, only one work attempted to exploit multimodal features of the RCPSP. Pérez et al. (2015) applied Multi-Modal Genetic Algorithm (MMGA) to solve the Resource-Constrained Multi-Project Scheduling Problem (RCMPSP), which is the derivative of the standard RCPSP. In their work, the authors were able to prove that multiple optima can be obtained in the RCMPSP, as well as to demonstrate that multimodal techniques provide better performance than other alternative commonly accepted methodologies for RCMPSP. Moreover, in his previous works (Pérez, Herrera, & Hernández, 2003; Pérez, Posada, & Herrera, 2012), Pérez successfully applied similar approach for solving the JSSP.

There are several explanations to the lack of enthusiasm among researchers in addressing these issues. On the one hand, optimisation of several objectives necessitates the development of an alternative optimisation and decision model that will differ from the standard formulations of RCPSPs. As a consequence, new means of the solution representation and schedule generation need to be developed. On the other hand, obtaining multiple global solutions requires the application of a special class of algorithms that are specifically designed for multimodal optimisation problems. Due to the discrete fitness landscapes of RCPSPs, and other combinatorial optimisation problems in general, the algorithms from that category are not typically applied to solve them. Hence, the application of such algorithms for RCPSPs induces the development of means of their porting.

The main goal of this PhD thesis is to propose an optimisation model for scheduling projects in a variable environment. In the proposed model, decisions regarding resource allocation and activity sequencing are simultaneously considered while taking into account the experience, efficiency and learnability of each resource type. In here, resource refers to human resource or members of the project team; resource efficiency reflects the speed at which an activity can be implemented by the project team; experience is defined as the total amount of time that members of the project team have previously spent on working in a similar problem; and learnability is the reflection of how quickly resource acquires its experience. To solve the optimisation problem, an evolutionary computation (EC) method that is capable obtaining multiple global optima is developed and several important aspects of its application to this problem as investigated, such as solution representation, difference estimation between solutions, and schedule generation.

In order to achieve this goal, the following objectives have been set:

- To define an optimisation model for scheduling in an uncertain environment:
    - Where activity durations are subject to influence of external factors;
    - Which considers optimisation of primary and secondary objectives;
    - Which takes the advantage of the RCPSP multimodal property.
- To develop different methodologies to solve the defined RCPSP problems:
    - To study, review and select the suitable solution representation and schedule generation schemes
    - To explore and extend advanced evolutionary computation techniques capable of global search to solve those problems
    - To explore techniques to obtain multiple solutions for RCPSPs
    - To evaluate the efficiency of the proposed algorithms in solving benchmark problems
    - To apply the proposed methodologies to solve a real RCPSP problem

1.2 Contributions

Throughout the work on this thesis, the following contributions to the science and engineering have been made:

- A new optimisation model for scheduling large-scale projects which takes into account the efficiencies and learnabilities of the resources (Chapter 3);
- Design of discrete cuckoo search (DCS) algorithm and its subsequent application to the RCPSP (Chapter 4);
- Design of a new discrete flower pollination algorithm (DFPA) is introduced and applied to solve the RCPSP (Chapter 4);
- Extension of DCS to derive an improved discrete cuckoo search (IDCS) algorithm is proposed and its application to solve the RCPSP (Chapter 4):
    - Paradigm of the original cuckoo search (CS) is changed and now uses crossover operator to create new individuals;
    - The algorithm operates on a novel solution representation scheme called event list (EL);
    - New crossover operator based on the EL is proposed. The operator is designed to combine useful problem-specific information extracted from the parent for the purpose of generating high-quality children.
- A new discrete species conserving cuckoo search (DSCCS) method able to obtain multiple global solutions of tackling multimodal properties of the RCPSP is introduced (Chapter 4):
    - Adaptation of species conservation technique and subsequent application to problems in the discrete domain;
    - Application of the species conservation technique to the IDCS

1.3 Publications

To this date, the following publications have been authored or co-authored based on the works done during this PhD:

Journal papers:

- K. Bibiks, Y. F. Hu, J.-P. Li, P. Pillai, A. Smith, “Discrete species conserving cuckoo search for the resource constrained project scheduling problem,” *IEEE Transactions on Evolutionary Computation*, **Submitted**
- K. Bibiks, Y. F. Hu, J.-P. Li, P. Pillai, A. Smith, “Improved discrete cuckoo search for the resource constrained project scheduling problem,” *Applied Soft Computing*, **Accepted, subject to revision**
- K. Bibiks, J.-P. Li, Y. F. Hu, “Discrete flower pollination algorithm for the resource constrained project scheduling problem,” *International Journal of Computer Science and Information Security*, vol. 13, no. 7, pp. 15-22, 2015

Conference papers:

- K. Bibiks, Y. F. Hu, J.-P. Li, A. Smith, “Discrete cuckoo search for the resource constrained project scheduling problem,” *IEEE 18th International Conference on Computational Science and Engineering*, Porto, 2015
- M. Amir, Y. F. Hu, P. Pillai, K. Bibiks, “Interaction Models for Profiling Assets in an Extensible and Semantic WoT Framework,” in *Proceedings of the Tenth International Symposium on Wireless Communication Systems*, Ilmenau, 2013

1.4 Organisation of the Thesis

The remainder of this thesis is organised as follows.

Chapter 2 contains literature reviews related to the most relevant scheduling problems and solution methods that were applied to solve these problems. Moreover, methodologies for application in multimodal scenarios are reviewed as well.

Chapter 3 details optimisation model of the problem that is considered in this thesis. Mathematical description of the problem is provided and similarities and differences with other scheduling problems in the literature are given.

Chapter 4 describes preliminary work that was done during the development of methodology for the proposed optimisation model. Several metaheuristic algorithms are presented. Performances of the developed algorithms are assessed using sets of benchmark instances from the PSPLIB, which are then compared with performances of other state-of-the-art heuristics. Sections of this chapter were used as foundations for research articles for publications in journals and conferences (see Section 1.3).

Chapter 5 presents case studies that are based on the optimisation model proposed in Chapter 3 and which are used to assess performance of the metaheuristic methodology presented in Chapter 4. The first case study is based on a real-life research project which consists of 51 activities and 4 types of resources and scheduling of which is subject to a number of uncertainties outlined in the chapter. Multimodal fitness landscape nature of the case study is exploited by the developed metaheuristic when a set of optimal solutions is obtained out which the most efficiently balanced one is chosen. For the second case study, several popular algorithms for the RCPSP are implemented and tested on the sets of benchmark instances that were modified to include additional parameters specific for the proposed optimisation model.

Finally, Chapter 6 concludes the work done in this thesis, outlines summary of achievement and contributions that were made and provides possible directions for future work.

Chapter 2 Literature Review

This chapter presents relevant project scheduling problems to this research and reviews literature on methods that were previously applied to solve them.

First, the Resource-Constrained Project Scheduling Problem (RCPSP) is introduced. Mathematical description of the problem is provided.

This is followed by an introduction of the Stochastic Resource-Constrained Project Scheduling Problem (SRCPSP), its mathematical description and fields of applied methodologies.

Lastly, the state-of-the-art metaheuristic methodologies are presented for each of the presented metaheuristics with examples of applications of these methodologies to relevant problems.

2.1 Resource-Constrained Project Scheduling Problem

Project scheduling addresses a problem of finding an optimal sequence of a set of activities that are associated with a set of resources such that all set objectives are optimised and all constraints are satisfied. It has been an active area of research for many decades and has drawn an increasing in recent years. Various scheduling problems have been studied in the literature (Weglarz, 1999; Kolisch & Padman, 2001), however, despite the varieties, all of these problems are NP-hard (Blazewicz, Lenstra, & Kan, 1983) combinatorial optimisation problems that can be modelled as variations of the RCPSP.

2.1.1 Mathematical Formulation

In the literature, large variety of the RCPSP derivatives exist (Hartmann & Briskorn, 2010), however, all of them, in one way or another, are based on the standard classical RCPSP. The main objective of standard classical RCPSP is to find an optimal schedule with minimal duration by assigning starting time to each activity in a project with respect to precedence relations and resource availabilities.

In the RCPSP a project is represented by a finite set of activities (i.e. jobs) $\boldsymbol{V} = \{0, 1, \ldots, n, n+1\}$. Activities 0 and n + 1 are unique dummy activities which represent the start and the end of a project, respectively. Typically, activities that constitute a project are represented by an activity-on-node (AON) network, also

sometimes referred to as project network, denoted as $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ (Zhou & Chen, 2002), where $\boldsymbol{V}$ is a set of nodes that denote activities and $\boldsymbol{E}$ is a set of arcs that denote precedence constraints. Alternatively, precedence constraints can be denoted as $i \longrightarrow j$ or (i, j). *Pred*(j) defined the set of direct predecessors, while *Succ*(j) is the set of direct successors of activity j. The processing time of activity j is given by p_j. The processing time of dummy activities 0 and $n + 1$ is $p_0 = p_{n+1} = 0$.

For their execution activities require renewable resources. In the context of the RCPSP, the term 'renewable resources' is defined as a pre-specified number of units of a resource being available for every period of the *planning time horizon* T (i.e. time period during which activity of going to be executed). Resources are defined by a finite set $\mathcal{R}^{\rho}$. The total availability of resource unit k is defined by R_k^{ρ}. The period usage of activity j of renewable resource k is denoted by r_{jk}^{ρ}, whereas the total resource consumption of renewable resource k by activity j is given by r_{jkm}^{ρ}.

The starting times of activities are represented by a schedule $\boldsymbol{S} = \{S_0, S_1, \ldots, S_n, S_{n+1}\}$, where S_j is the starting time of activity j. S_0 is used as a reference point which signifies the start of a project and is always assumed to be 0. Consequently, set $\boldsymbol{C} = \{C_0, C_1, \ldots, C_n, C_{n+1}\}$ denotes completion times, where C_j is completion time of activity j. The total duration of a project, or its makespan, will be equal to the completion time of the last activity C_{n+1}. $\mathscr{L}_{\mathrm{T}}$ defines the set of time-feasible schedules, $\mathscr{L}_{\mathrm{R}}$ the set of resource-feasible schedules and $\mathscr{L} = \mathscr{L}_{\mathrm{R}} \cap \mathscr{L}_{\mathrm{T}}$ the set of feasible schedules. Finally, d_{ij}^{min} and d_{ij}^{max} denote minimum and maximum time lags, respectively, between the start of activities i and j.

Taking into consideration the above-presented formulation, the optimisation model of the RCPSP can then be stated as follows:

$$PS \mid prec \mid C_{\max} \tag{1}$$

where PS stand for project scheduling, $prec$ signifies precedence feasible and $Cmax$ represents completion time of last activity in the project. This model forms the core problem among the class of standard deterministic RCPSPs and it means that while minimising the project's makespan, precedence and resource constraints needs to be observed.

2.1.2 Applied Methodologies

Due to the simplicity of the definition, wide applicability, and high complexity, the RCPSP has attracted a considerable amount of attention from researchers and vast amount of methodologies has been proposed for solving it. Blazewicz et al. (1983) showed that, as a generalisation of the classical Job-Shop Scheduling Problem (JSSP) (Graham, 1966), RCPSP belongs to a class of NP-hard combinatorial optimisation problems (Leeuwen, 1998).

Kolisch and Hartmann (1999) did a comprehensive review of different methods proposed to solve RCPSP and classified them into two categories: exact methods and heuristic approaches, whereas heuristic approaches were further divided into priority rule-based methods and metaheuristics. In the last 20 years both exact solution procedures and heuristics have witnessed a tremendous growth and improvement, which is confirmed by the amount of published surveys (Hartmann & Kolisch, 2000; Kolisch & Hartmann, 2006; Herroelen W. , 2006; Ozdamar & Ulusoy, 1995; Demeulemeester & Herroelen, 2002).

In order to evaluate algorithms proposed for the RCPSP and create basis for comparison, Kolisch and Sprecher (1997) proposed to measure performance of the algorithms by applying them to schedule benchmark isntances from Project Scheduling Problem Library (PSPLIB). The following setup has been proposed. Three sets of benchmark instances need to be used: 480 instances from J30 set, each of which consisted of 30 activities and 4 resources; 480 instances from J60 set, each of which consisted of 60 activities and 4 resources; and 600 instances from J120 set, each of which consisted of 120 activities and 4 resources. In order to pass the evaluation, the algorithm has to be applied to schedule all instances from J30, J60, and J120 sets. The only compulsory criterion in this setup is the setting of stopping criterion, value of which needs to be 1000, 5000 and 50000 objective evaluations. After running each of the instances, performance of the algorithm is measured by calculating average deviation percentage from Critical Path (CP). Computational time in this evaluation is not taken into account is it will largely depend on the experimental setup. Such way of measuring performance of the RCPSP methodologies has become de facto standard way of evaluation and it has been adopted by many researchers.

Tormos and Lova (2001) in their research tested several popular exact solution procedures and heuristics for the RCPSP. Performances of the selected algorithms were measured using the setup proposed by Kolisch and Sprecher

(1997). The most competitve exact algorithms were the ones of Brucker et al. (1998), Mingozzi et al. (1998), and Specher (2000). Nevertheless, even though these exact algorithms demonstrated good performances, in a satisfactory manner they were only capable of solving small-scale instances of problems with up to 60 activities.

Another comprehensive survey done by Hartmann and Kolisch (2000) and its update version (Kolisch & Hartmann, 2006) provided a classification and performance evaluation of different state-of-the-art heuristics that have been proposed for RCPSPs. As was shown by their experimental evaluation, metaheuristic methods demonstrate far better performance than heuristics. For 53 methods sorted with respect to the performance of evaluation 1000, 5000, and 50000 schedules, the best methods for J30, J60 and J120 sets were all metaheuristic approaches, which included genetic algorithm (GA) with path relinking (Kochetov & Stolyar, 2003), scatter search (SS) (Debels, De Reyck, Leus, & Vanhoucke, 2006), hybrid GA (Valls, Ballestin, & Quintanilla, 2003), and simulated annealing (SA) (Bouleimen & Lecocq, 2003).

Because of this, as of today, the application of metaheuristic algorithms is considered to be the most effective and reliable way of solving the RCPSP.

2.2 Stochastic Resource-Constrained Project Scheduling Problem

Most of the literature on the project scheduling concentrates on finding a schedule with fixed activity durations and starting times, which is then used as a guideline for the actual execution of a project. In the real-world, however, during the execution of a project unexpected events or circumstances can cause deviations from the original schedule. Examples of such can include an under- or overestimation of the workforce, cancelations, delays based on unexpected issues, equipment failure, nature disasters, etc. In the majority of the situations, these kinds of events can be modelled as fluctuations in the activity durations.

In comparison to deterministic project scheduling, little research on project scheduling under risk and uncertainty can be found in the literature. Herroelen and Leus (2005) reviewed various problems related to this field and identified the most important research tracks in this area:

- *Reactive Scheduling*

- *Proactive Scheduling*
- *Stochastic Project Scheduling*
- *Fuzzy Project Scheduling*

Reactive scheduling deals with the uncertainties in scheduling by revising or re-optimising the baseline schedule when an unexpected event happens. Actions that are taken during such revisions may be based on various underlying strategies. On the one hand, the reactive approach may rely on very simple techniques such as schedule repair action or right shift rule. Paradigms of these techniques were first introduced by Sadeh et al. (1993) and Smith (1994), respectively. These approaches work by moving forward in time all activities that were affected by the schedule breakdown. Moving of activities is done either because they were using resources that caused the breakage or because of the precedence relations. Since such strategy does not re-sequence activities, it may lead to poor results. On the other hand, the reactive scheduling approach may involve full rescheduling of the affected part of the schedule that remains to be executed. Such approach is commonly referred to as rescheduling. The goal of rescheduling is to generate a new schedule that will deviate from the original one as little as possible. As performance measure, the new project makespan is used. Such strategy may rely on the use of exact or heuristic algorithms that use the minimisation of the sum of the difference between the activities' starting times in the original and repaired schedules as the objective (El Sakkout & Wallace, 2000). Calhoun et al. (Calhoun, Deckro, Moore, Chrissis, & Van Hove, 2002) used goal programming to revise project schedule with the initial objectives and the objective of minimising the number of changed activities. Another way of dealing with deviation from the initial activity duration projection is via application of match-up scheduling practices (Bean, Birge, Mittenthal, & Noon, 1991).

Proactive scheduling (also known as robust scheduling), in comparison to reactive scheduling, considers future disruptions during initial schedule generation. The main concern here is the generation of initial schedule that will minimise the effects of disruptions on the performance measures, therefore robustness is considered as the primary objective (Mehta & Uzsoy, 1998). Several robustness measures exist. Some robustness measures are based on the actual performance of the realised schedules, and some are based on regrets. The regret represents the difference between performances of realised and optimal schedules (Demeulemeester & Herroelen, 2011). Several techniques

for achieving robustness exist, such as fault tolerance (Ghosh, 1996), temporal protection (Gao, 1995), time window slack (Davenport, Gefflot, & Beck, 2001), minimax objective (Sevaux & Sorensen, 2002) and abstraction of resource usage (Herroelen & Leus, 2004).

Stochastic project scheduling, or more commonly known as stochastic resource-constrained project scheduling problem (stochastic RCPSP or SRCPSP), aims at scheduling project activities with known activity duration distribution. The main objective in the SRCPSP is the minimisation of the expected project makespan subject to precedence relations and limited resource capacities. However, unlike in the deterministic RCPSP, the outcome of the SRCPSP is a so-called scheduling policy (Mohring, Radermacher, & Weiss, 1984). The execution of a project in the SRCPSP represents a multi-stage decision process, where at each consecutive step, by acting as a scheduling rule, policy determines which activity is going to be started next.

Lastly, fuzzy project scheduling assumes that probability distributions for the activity durations are unknown due to the lack of historical data. In situations that involve imprecision rather than uncertainty, the use of fuzzy numbers for modelling activity durations is recommended. The outcome of a fuzzy scheduling pass normally is a fuzzy schedule, which indicates fuzzy starting times of the activities in the project. Dorn et al. (1995) noted that a fuzzy schedule at certain levels gives some degree of freedom and lets to choose the starting times of certain activities a little earlier or later when soft constraints may be imposed. In this sense, a fuzzy schedule consists of multiple crisp schedules. The most recent work on fuzzy scheduling has been gathered by Slowinski and Hapke (2010).

Herroelen and Leus (2005) outlined advantages of the SRCPSP over other research tracks and its higher suitability for the project scheduling under uncertainties, mainly for the reason that it does not require a generation of the baseline plan for making advance commitments to both subcontractors and customers. Moreover, the authors also highlighted SRCPSP's similarity with its deterministic variant and confirmed the possibility of application of some of the RCPSP methods for the SRCPSP, given necessary modifications are made.

2.2.1 Mathematical Formulation

Similar to the deterministic RCPSP, the main goal of the SRCPSP is to minimise the expected project makespan while considering limited resource availabilities and precedence relationships.

Following the notation of the RCPSP presented in Section 2.1.1, a project in the SRCPSP is similarly represented by an AON graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ (Zhou & Chen, 2002), where $\boldsymbol{V} = \{0, 1, 2, \ldots, n, n + 1\}$ denotes the set of activities and E is a set of arcs representing zero-lag finish-start precedence relations. In the SRCPSP, activities 0 and $n + 1$ are assumed to be dummy activities that represent the start and end of the project, respectively. The durations of other activities are denoted by a random vector $\boldsymbol{d} = (d_1, d_2, \ldots, d_n)$, where d_j denotes the random duration of activity j.

For their execution, activities require renewable resources which are defined by a finite set $\mathcal{R}^{\rho}$. The total capacity of resource k is denoted by R_k^{ρ}. Activity j requires an amount of $r_{jk}^{\rho} \leq R_k^{\rho}$ units of resource type k.

Given the presence of both resource and precedence constraints, schedules are generated through application of so-called scheduling policies or strategies.

According to Olagu´ibel and Goerlich (1993), scheduling policy Π makes a decisions at decision point t, where decision at time t is to start at time t a *precedence and resource feasible set of activities* $S(t)$, where feasible means that all constraints are respected. The decision may only exploit information that is available until the current time t. As soon as the execution of activities is complete, their final durations becomes known, yielding to realisation of d. The application of policy Π leads to the creation of a schedule $\Pi(d) = \{S_0, S_1, S_2, \ldots, S_n, S_{n+1}\}$ of activity start times and the resulting schedule makespan $C_{max}(\Pi(d))$.

Therefore, optimisation problem of the SRCPSP can be stated as creation of scheduling policy that will minimise the expected project duration $\boldsymbol{E}[C_{max}(\Pi(d))]$ over a class of policies. If it is assumed that the distributions of the activity durations are discrete, then the problem is a generalisation of the deterministic RCPSP presented in Section 2.1.1, therefore its difficulty can be described as NP-hard in the strong sense (Demeulemeester & Herroelen, 2002).

2.2.2 Applied Methodologies

The literature on relevant methods proposed for solving the SRCPSP is rather limited. All presented approaches can be divided into two categories: on the one

hand, there are theoretical studies and applications of general or particular classes of scheduling policies; on the other hand, there are various classes of heuristics proposed.

2.2.2.1 Scheduling Policies

A scheduling policy can be regarded as a dynamic decision process that decides which activity is going to be started at the current decision time t. Various classes of scheduling policies have been proposed for the SRCPSP. Stork (2001) did a comprehensive review of existing policies and summarised them as follows:

- *Priority policies*
- *Pre-selective policies*
- *Non-anticipativity constraint*
- *Earliest start policy*
- *Linear pre-selective policy*

First introduced by Radermacher (1981), a priority policy is the best-known class of scheduling policies for the SRCPSP. A policy is called priority if at decision time t a maximum amount of available activities is scheduled according to their respective priority numbers. Due to their properties, priority policies are easy to define and implement, however, at the same time, applications of such policies can result in an optimal schedule not being found. Furthermore, a deviation in the activity processing times may cause specific anomalies to appear, resulting in prolonged schedule duration, even though the activity processing times have been reduced. In the literature, such behaviour is commonly referred to as *Graham anomaly* and it has been described by Graham (1966) in the context of parallel machine scheduling. For example, such events as decrease in the duration of activities, addition of additional capacity and removal of precedence constraints may lead to increase of the project makespan. As the result of Graham anomalies, priority policies are very rarely used for the SRCPSP.

Radermacher (1984) introduced another class of policies called pre-selective policies. These policies work by establishing sets of preselected activities called minimal forbidden sets. The execution of these sets is then postponed until at least one activity from the defined set has been completed. In contrast to priority policies, preselective policies are not susceptible to Graham anomalies, hence are more robust. The preselective policies were further studied by Igelmund and

Radermacher (1983). Mohring and Stork (2000) introduced quite useful representation of pre-selective policies using so-called waiting conditions. Waiting conditions are modelled as AND/OR precedence constraints.

In various publications on stochastic scheduling problems (Wets, 1989; Escudero, Kamesam, King, & Wets, 1993) another class of scheduling policies appears, called non-anticipativity constraint. The requirement of a non-anticipativity constraint is that decision made at any decision time can only be based on the information available at that moment. Radermacher (1985) has shown that this requirement exactly expresses the possibility of using the maximal amount of available information for each decision point.

Rockafellar and Wets (1991) presented a class of robust policies, which includes earliest start policy among all. The earliest start policy is closely related to the class of preselective policies and can be viewed as a pair of a combinatorial object and an algorithm which transforms a given scenario into a schedule.

Lastly, Stork (2001) proposed linear preselective policies which combine the list-oriented features of priority policies with the selection-oriented character of preselective policies. The idea is to define the selection via a priority ordering of the activities.

Since scheduling procedures rely on a preliminary enumeration, the use of scheduling policies for the SRCPSP becomes computationally intractable when the size of projects reaches to practical instances (e.g. 50 activities or more). Stork (2001) concluded that for larger instances the only remaining alternative to the scheduling policies is the application of heuristic approaches.

2.2.2.2 Heuristics

There are very few heuristics developed for the SRCPSP. One of the first computational methods proposed to address the issue of uncertain activity durations in project scheduling was the project evaluation review technique (PERT) (Malcolm, Roseboom, Clark, & Fazar, 1959). PERT analysis works by estimating the expected project duration and predicts possible deviations from the baseline schedule, assuming probability distributions of activity durations are known. Dodin (2006) reviewed different variations of PERT methodologies and outlined that main limitation of these methods is the lack of decision-support. This limitation is caused by the fact that PERT methodologies only focus on understanding the statistical properties of a project makespan, yet they do not

identify the optimal activity starting times, nor identify which path(s) of the schedule will be critical. Several attempts have been made to address this limitation (Dodin, 1984; Elmaghraby, Ferreira, & Tavares, 2000), however, the line of this research did not produce any significant results, mainly due to the fact that proposed methods do not explicitly use resource constraints, instead resources are assumed to be unlimited.

More recent research tracks on the SRCPSP have been dealing with the application of various metaheuristics. Two-phase genetic algorithm (GA) (Ashtiani, Leus, & Aryanezhad, 2011). The method useD a pre-processing procedure to estimate a sequence of all activities at time zero, skipping the observation of early activities. In the optimal control theory, such terminology corresponds to an open-loop policy (Martinez & Soares, 2002). Typically, methods based on this policy are static in nature and during the execution of the project they are not updated.

In contrast to open-loop policies, an alternative solution is a closed-loop policy. The main difference between the two is that closed-loop policy makes the scheduling decisions in a dynamic fashion through the application of the dynamic programming (DP) (Bertsekas, 2007). Instead of scheduling an entire activity sequence for the whole project, a closed-loop policy at each decision point selects activities that are permissible for a start. This selection is based on an optimal decision rule and is made by the decision-maker, given the relevant information about the project is known. Several DP-based approaches for the SRCPSP have been proposed in the chemical-pharmaceutical environment (Choi, Realff, & Lee, 2004). Another DP approach was presented by Haitao and Womer (2015). The authors presented an efficient and effective approximate DP algorithm based on the priority policy. The performance of the algorithm was enhanced by employing constraint programming, which subsequently improved the performance of base policy offered by a priority rule-based heuristic.

2.3 Methodologies

As can be noted from the surveys on the presented problems (Kolisch & Hartmann, 1999; Kolisch & Hartmann, 2006; Herroelen & Leus, 2005), the most effective algorithms proposed for solving them belong to the class of metaheuristics. The use of metaheuristics for application in scheduling problems

(and optimisation problems in general) is a rapidly growing area of research. Due to the importance of these problems for the scientific as well as engineering worlds, each year, more and more innovative methodologies are being proposed. Dreo et al. (2006) presented a survey of nowadays most important metaheuristic methodologies from a conceptual point of view. The authors analysed differences and similarities of all known metaheuristics and outlined their concepts and components. All methodologies there are covered in the survey were classified as follows:

- *Nature-inspired* or *non-nature inspired*
- *Population-based* or *single point search*
- *Dynamic* or *static objective function*
- *One* or *various neighbourhood structures*
- *Non-hybrid* or *hybrid*
- *Single* or *multiple solutions*
- *Memory* or *memory-less methods*

Clearly, metaheuristic algorithms are not restricted to only one classification, hence one method can be classified to belong to several groups. From the research point of view, the most significant category of metaheuristics is *population-based* vs *single point search* (Blum & Roli, 2003). This is explained by the fact, that nowadays the biggest trend in the field of engineering optimisation is a hybridisation of methods: integration of single point search methodologies into population-based ones. Thus, metaheuristics from these categories are used as a base for forming more advanced and complicated methods. Nevertheless, such categories of metaheuristics as *hybrid* and *multiple solutions* are not least important, as these methodologies usually represent state-of-the-art solutions for the most complicated optimisation problems.

2.3.1 Single Point Search Algorithms

The search process of single point search methods (or more commonly known as *trajectory methods*) is characterised by a trajectory in the search space, meaning that the next solution found by the algorithm may belong to the neighbourhood of the previous solution. Hence, the search process of these algorithms can be viewed as the evolution of a dynamical system in time (Bar-Yam, 1997). The process of operation of these algorithms typically begins with an initial solution represented by an initial trajectory in the search space, which is

constantly improved via solution improvement mechanism until predefined stopping condition is met. The most commonly used single point search metaheuristics are *simulated annealing* (SA) (Kirkpatrick, Gelatt, & Vecchi, 1983), *tabu search* (TS) (Glover, 1986), *greedy randomised adaptive search procedure* (GRASP) (Feo & Resende, 1995), *variable neighbourhood search* (VNS) (Hansen & Mladenovic, 1999), *guided local search* (GLS) (Voudoris & Tsang, 1999), and *iterated local search* (ILS) (Martin, Otto, & Felten, 1991).

2.3.1.1 Simulated Annealing

Among all single point search metaheuristics, Simulated Annealing (SA) is considered to be the oldest one. First presented by Kirkpatrick et al. (1983), its fundamental idea is to allow moves that will result in solutions of worse quality than the current best with the aim of escaping from the local optima trap. The process of SA begins with the creation of an initial solution (randomly or heuristically) and then its further improvement via local search. In the nutshell, this whole process can be characterised as a *Markov chain* (Feller, 1968), as it follows a trajectory in the state space where the choice of the next state only depends on the previous one. Because of that, the basic versions of SA are memory-less and can easily be integrated into other metaheuristics. Nevertheless, the inclusion of memory can be beneficial for the development of more advanced SA approaches (Chardaire, Lutton, & Sutter, 1995).

For the RCPSP, SA has been applied successfully a number of times. Boctor (1996) demonstrated fairly good performances of SA approaches on PSPLIB benchmark instances.

Bouleimen and Lecocq (2003) proposed an SA in which predictable search pattern is replaced by a new strategy that takes into account the properties and characteristics of the RCPSP solution space. Jozefowksa et al. (2001) applied the SA to solve the multi-mode variant of the RCPSP.

Nikulin and Drexl (2010) used SA to solve the airport flight gate scheduling problem which was modelled in the form of RCPSP. While previous approaches have been simplified to a single objective counterpart, they used SA to optimise more than one objective. The objectives were modelled by means of fuzzy members.

Apart from the RCPSP, SA has also been applied to several other combinatorial optimisation problems, such as Quadratic Assignment Problem

(QAP) (Connolly, 1990) and job-shop scheduling problem (JSSP) (Laarhoven, Aarts, & Lenstra, 1992). For more examples of the SA applications refer to (Aarts & Lenstra, 1997).

2.3.1.2 Tabu Search

Tabu Search (TS) is believed to be the most cited and commonly used. The concept of TS, which is based on the work of Glover (1977), was first introduced by Glover (1986). In comparison to other single point search methods, TS works by explicitly using a history of the search process. With this, TS can avoid falling into local optima trap as well as implement explorative strategies. As a basic ingredient for the solution search, TS applies the best improvement local search operator, whereas to escape from the local optima, TS uses a short-term memory, which is implemented in a form of tabu list. Tabu list keeps track of the previously visited neighbourhoods and forbids moves toward them. Therefore, the moves of TS are only permitted towards solutions neighbourhoods of which have not yet been visited. At each iteration, based on the tabu list, via improvement local search TS forms a set of allowed solutions. After the set is formed, the best solution from this set is chosen as a current best one. Due to the dynamics of the search process, TS is considered to be a dynamic neighbourhood explorative method (Stutzle, 1999).

Thomas and Salhi (1998) were the first to apply TS for the RCPSP. The most notable characteristic of the presented method is the utilisation of two diverse neighbourhood structures.

Similarly, Nonobe and Ibaraki (2002) proposed a TS method for the RCPSP which operates on the Activity List (AL) solution representation scheme and uses specific rules for exploring solution search space and defining structure of the neighbourhood.

Klein (2000) developed a so-called *reactive TS* for the RCPSP with time-varying resource constraints. The algorithm is based on the AL representation and uses a serial Schedule Generation Scheme (SGS) to convert solution representation scheme into a schedule. The solution search space is explored by swap moves, which involve the relocation of predecessors or successors of the swapped activities, given all precedence constraints are satisfied.

Another implementation of the TS for project scheduling was presented by Pan et al. (2009). The original TS model is improved by optimising the neighbourhood exploration mechanism.

Tsai and Gemmill (1998) developed a TS methodology for deterministic and stochastic variants of the RCPSP. To provide more diversified search, the presented adaptation of the TS that uses several tabu lists, randomised short-term memory, and multiple starting schedules.

One of the most recent applications of the TS for the RCPSP was done by Artigues et al. (2003). Their algorithm uses various insertion rules to estimate the structure of the neighbourhood and successfully explore the solution search space.

Moreover, TS has been applied to most of the other combinatorial optimisation problems, such as QAP (Taillard, 1991), maximum satisfaction (MAXSAT) problem (Battitti & Protasi, 1997), assignment problems (Dell'Amico, Lodi, & Maffioli, 1999), JSSP (Nowicki & Smutnicki, 1996), and the vehicle routeing problem (VRP) (Gendreau, Laporte, & Potvin, 2001).

2.3.1.3 Greedy Randomised Adaptive Search Procedure

First introduced by Feo and Resende (1995), Greedy Randomised Adaptive Search Procedure (GRASP) is a simple metaheuristic that has been created by combining a constructive heuristics and local search operator. The solution search process of GRASP consists of two phases: solution construction and solution improvement. The first phase, namely solution construction, is responsible for building a solution on a step-by-step basis by adding one element at a time. The mechanism responsible for this consists of two components: dynamic constructive heuristic and randomisation. The second phase of the algorithm is a basic local search process.

The basic versions of GRASP do not require to use any history of the search process. Therefore, the only memory requirement of this algorithm is to store the best solution found so far. This fact serves as the main reason for integration of GRAPS into other metaheuristics. Moreover, due to its simplicity, the GRASP is computationally fast and is capable of producing satisfactory results in a short amount of time.

In the literature, there are not many examples of the application of GRASP for the RCPSP. The most prominent application of GRASP was done by Alvarez-

Valdes et al. (2008). The authors studied a generalisation of the classical RCPSP which considers the new type of resource. For this problem, several pre-processing techniques are developed which help to determine the existence of feasible solutions and reduce the number of variables and constraints. The developed techniques are applied in conjunction with the GRASP.

Another application of the GRASP is proposed by Ballestin and Leus (2009), this time for the SRCPSP. The authors developed an integrated simulation-optimisation framework, in which GRASP is applied to explore the solution search space, while the simulation is used to evaluate the solutions found in the local neighbourhood.

Applications of the GRASP for other optimisation problems were proposed by Binato et al. (2000) for the JSSP, Resende and Ribeiro (2001) for the graph planarization problem (GPP), and Prais and Ribeiro (2000) for assignment problems. For more examples of GRASP applications refer to (Festa & Resende, 2002).

2.3.1.4 Variable Neighbourhood Search

Variable Neighbourhood Search (VNS) is a metaheuristic proposed by Hansen and Mladenovic (1999). Its idea is based on the application of strategy that is based on a dynamically changing structure of the neighbourhood. The concept of the algorithm is very general and there are many degrees of freedom for designing its variants and instantiations. The basic versions of VNS begin its operation by the structure of the neighbourhood and generating the initial population. After this is done, the main cycle begins, which is composed of three phases: shaking, local search, and move. In the shaking phase a solution $s`$ in the k^{th} neighbourhood of the current solution s is randomly selected. The randomly selected solution then becomes the local search starting point. The local search then explored other neighbourhoods for a new solution. At the end of the local search process, the new solution $s``$ is compared with s and, if it is better, replaces it and the algorithm starts again with $k = 1$. Otherwise k is incremented and a new shaking phase is initiated.

Fleszar and Hindi (2004) were the first to propose to solve the RCPSP by applying the VNS method. The proposed method uses enhanced move operator which relocates activities within the AL. To speed up the exploration of the search space, authors apply specific lower bounds calculations.

Another application of the VNS for the RCPSP is done by Kochetov and Stolyar (2011). The developed algorithm includes a new local search and moves operators, which greatly improve the performance of the method in comparison to other implementations of the VNS for the RCPSP. The method is tested on a set of benchmark instances. The results of testing are satisfactory; the algorithm shows competitive performance against other trajectory metaheuristics.

Roshanaei et al. (2009) developed a variation of the VNS for the RCPSP. The fundamental difference of the presented method is the obviation of the notorious chaotic behaviour of local search-based metaheuristics by the means of insertion of several systematic neighbourhood structures.

Moreover, other notable applications of the VNS and its variants for other optimisation problems were done by Fleszar et al. (2009) for the VRP, Fonseca and Santos (2014) for high-school timetabling, and by Abdelmaguid (2015) for the JSSP.

2.3.1.5 Guided and Iterated Local Searches

In contrast to other trajectory methods, which deal with the static neighbourhood, the Guided Local Search (GLS) (Voudoris & Tsang, 1999) and Iterated Local Search (ILS) (Martin, Otto, & Felten, 1991) dynamically change the structure of the neighbourhood space to provide more efficient and effective exploration. GLS explores the search space by dynamically changing the objective function, while ILS does it with the means of the perturbation (i.e. finds local optima, perturbs the solutions, and then restarts the process). Both of these variations of the local search most of the times serve as a framework for other metaheuristics or are integrated directly into them.

For the RCPSP, local search methods are usually used in conjunction with other algorithms. Therefore, in the literature, there are not many examples of the application of non-hybrid local search metaheuristics.

A local search algorithm for the RCPSP was developed by Pesek et al. (2007). Their method uses several improved neighbourhood structures that are identified by relocating a fixed amount of activities in the AL.

Palpant et al. (2004) presented a local search strategy in which a subpart of the current solution is fixed and the other part defines a sub-problem which is solved by applying a heuristic or an exact method.

A complicated local search strategy for the RCPSP is presented by Ranjbar (2008). The method represents a filter and fan methodology which operates on the AL solution representation and consists of two major mechanisms: local search; and filter and fan strategy. The local search uses local move operators to estimate structure of the neighbourhood, while the filter and fan strategy uses a list of obtained local optima to evaluate the solution neighbourhood defined by the local search.

Nevertheless, GLS, ILS and its variations were extensively applied to solve other optimisation problems. For example, Mills and Tsang (2000) applied GLS to solve the weighted MAXSAT, Kilby et al. (1999) used GLS to solve the VRP, Voudouris, and Tsang (1999) developed a variation of the GLS for the TSP. Further, ILS was applied to the TSP (Martin, Otto, & Felten, 1991), QAP (Lourenco, Martin, & Stutzle, 2001), and the single machine total weighted tardiness problem (Besten, Stutzle, & Dorigo, 2001).

2.3.2 Population-Based Algorithms

Rather than dealing with a single solution, population-based methods at every iteration deal with a set (i.e. population) of solutions (i.e. individuals). Because of this property, population-based metaheuristics explore the search space in a more natural and intrinsic way, while the final result of the operation strongly depends on the way the population is manipulated. The most studied group of population-based metaheuristics in the literature is the category of Evolutionary Computation (EC) (Back, Fogel, & Michalewicz, 1997). EC is a sub-field of algorithms that are inspired by the nature and capability of living beings to evolve and adapt to their environment. Therefore, the algorithms from that category can be characterised as computational models of the evolutionary process.

2.3.2.1 Genetic Algorithm

Genetic Algorithm (GA), inspired by the process of biological evolution, has been introduced by Holland (1975) and is one of the most widely used metaheuristics. The ideas of the GA are based on the survival of the fittest process when over consecutive generations individuals evolve and the strongest among them survive. Each generation consists of a population of individuals in which each individual represents a point in the search space and a possible solution to the problem at hand. Individuals in the population are made to go through a process of evolution which consists of selection, mating (crossover) and mutation.

Husbands et al. (1996) outlined the advances of GAs for scheduling and illustrated the resemblance between scheduling and sequence-based problems.

One of the first implementations of GA for the RCPSP was presented by Hartmann (1998). He proposed to use a variation of GA where every gene composing a chromosome is a delivery rule. Later, Hartmann (2002) improved his original method by introducing a self-adapting mechanism. With the inclusion of this mechanism, the algorithm is capable of adapting to the problem instance by learning which of the decoding procedures is more successful.

Alcaraz and Maroto (2001) proposed an improved version of the GA which includes a new solution representation scheme and advanced crossover technique. Further, Mori and Tseng (1997) have applied it to solve the multi-mode variant of RCPSP.

Coelho and Tavares (2003) presented a GA which is based on the AL representation scheme and serial SGS. The authors proposed a new crossover operator for the AL called late join function crossover. The operator constructs new individuals by adopting the solution of the first parent and swapping each adjacent pair that is in reverse order in the second one. Similarly, the implementation of GA for the RCPSP developed by Hindi et al. (2002) is also based on the AL representation and serial SGS. However, in this method, the initial population is produced by a pure random mechanism.

Tolku (2002) developed a GA which instead of using solution representation schemes, directly works with schedules (i.e. vectors of starting times). Since genetic operators (crossover and mutation) may produce infeasible schedules, the author developed a penalty function, which is used to evaluate the constraint violations.

A slightly different approach for GA implementation was shown by Zhu et al. (2011). Unlike any other traditional GA implementations, the authors implemented a resource fragment mechanism that stores such information as starting times and resource distributions of the activities. In order to generate a feasible schedule and further improve it, a resource allocation and schedule enhancement methods are implemented. The quality of the constructed schedule is evaluated with the new fitness function.

GAs have been extensively applied to most of the optimisation problems. Examples of such uses include the TSP (Grefenstette, Gopal, Rosmaita, & Gucht, 1985) and QAP (Kratica, Tosic, Filipovic, & Dugosija, 2011) among all. For an

extensive collection of more examples of GA applications refer to the survey of Blum and Roli (2003).

2.3.2.2 Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) was introduced by Kennedy and Eberhart (1995) and represents a population-based stochastic optimisation technique inspired by social behaviour of bird flocking or fish schooling. In a PSO, at each iteration, a group of individuals is adjusted closer to the fittest member of the population. This principle resembles a flock of birds who circle over an area where they can smell a hidden source of food. The one who is closest to the food chirps the loudest and the other birds swing around in his direction. If any of the other circling birds comes closer to the target than the first, it chirps louder and the others move toward him. This tightening pattern continues until one of the birds happens upon the food.

One of the most notable applications of the PSO for the RCPSP was done by Chen et al. (2010). The authors proposed two scheduling rules: delay local search and bi-directional scheduling rule. To speed up the procedure of solution makespan evaluation, critical path calculations are used. Simulation results indicate the efficiency of the proposed algorithm in producing high-quality solutions.

Another application of the PSO for the RCPSP was presented by Zhang (2005). In order to deal with permutation feasibilities and precedence constraints during particle flying, the hybrid particle-updating mechanism is introduced. The potential solutions represented by particles (or exact positions of the particles) either in the permutation form or in the priority form are transformed to feasible schedules. The transformation is done using a serial SGS.

Linyi (2007) presented an implementation of a PSO for the RCPSP with the one-point crossover. The author introduced a new method for calculating the crossover function based on the precedence feasibility list of the activities. Moreover, a new approach for representation of particle evolution is presented and explained. The computational experiments showed that the developed modification of the PSO outperformed several other non-hybrid heuristics.

Moreover, applications of the PSO to other combinatorial optimisation problems are quite common. Few examples of such are TSP (Shi, Liang, Lee, Lu, & Wang, 2007) and JSSP (Surekha, Raajan, & Sumathi, 2010).

2.3.2.3 Ant Colony Optimisation

Ant Colony Optimisation (ACO) is a population-based metaheuristic approach proposed by Dorigo (1999), the main source of inspiration of which is the foraging behaviour of the real ants. ACO is based on a parametrised probabilistic model – the pheromone model – which is used to model the chemical pheromone trails. In the ACO, the solutions are incrementally constructed by ants. This is achieved by adding opportunely defined solution components to a partial solution under consideration. For this, artificial ants perform randomised walks on a completely connected graph, vertices of which are solutions components. In the literature, this graph is commonly referred to as the construction graph.

The first application of the ACO for the RCPSP was presented by Merkle et al. (2002). In the proposed approach, a single ant corresponds to one application of the serial SGS. The eligible activity to be scheduled next is selected using a weighted evaluation of the latest start time priority rule and so-called pheromones, which represent the learning effect of previous ants. A pheromone value describes how promising it seems to put a certain activity in the schedule. Further features of the approach include separate ants for forward and backwards scheduling and 2-opt-based local search phase at the end of the heuristic's operation.

An improved ACO for the RCPSP was introduced by Luo et al. (2003). The general ACO is improved by using the ant with backtracking capabilities and several kinds of heuristics for the construction of the solution. The combination of direct and summation pheromone evaluation methods and the pseudo-random-proportional action choice rule are also used.

Similarly, another implementation of the ACO was developed by Yuan et al. (2009). In the method, task duration and resources are considered as the heuristic information. This information is later used to calculate the accurate state transition probability and reach the scheduling optimisation.

Successful adaptations of the ACO to other optimisation problems include the application to routeing in communication networks (Caro & Dorigo, 1998), sequential ordering problem (SOP) (Gambardella & Dorigo, 2000), JSSP (Dorigo & Stutzle, 2003).

2.3.2.4 Scatter Search

Scatter Search (SS) and its more generalised form path relinking were developed by Glover et al. (2000). Their main difference from other ECs is the introduction of unifying principles for joining (or recombining) solutions based on the generalised path constructions in the Euclidean or neighbourhood spaces. Moreover, these methodologies also incorporate ideas that were originated from the TS, such as the use of adaptive memory and associated memory-exploiting mechanism. SS (as well as path relinking) is a search strategy that generates a set of solutions that are chosen from a set of reference solutions corresponding to the problem under consideration. Once the set of solutions is constructed, an improvement mechanism is applied. The improved solutions then form a set of dispersed solutions, which later is used as reference solutions at the next iterations.

One of the first application of the SS for the RCPSP was done by Ranjbar and Kianfar (2009). The authors presented an improvement to their original local search method, this time, it was SS method with double justification technique. The method operates on the topologically ordered AL representation and uses double justification to perform backwards and forward shifts for further improvements of the schedules. In the same year, another SS variant was proposed by Mahdi-Mobini et al. (2009). Similarly to the previous method, the presented algorithm also operates on the AL representation and uses double justification technique. On the other hand, this approach incorporates two point crossover operator, a path relinking strategy, and a permutation-based operator.

Another application of the SS for the RCPSP was proposed by Berthaut et al. (2014). The presented SS includes a new move operator and path relinking method which are used in conjunction together. The algorithm is tested on the PSPLIB test instances and demonstrated the best results among other compared metaheuristics.

The most recent SS adaptation for the RCPSP is developed by Paraskevopoulos et al. (2012). Unlike other methodologies for the RCPSP, the presented approach is based on a new representation of a solution called the event list. This representation scheme is based on an ordered list of events that are sets of activities that start at the same time. Moreover, the algorithm incorporates a new improvement method and event list-based combination method.

Because of its relative newness, there has been an increasing interest in SS in recent years. Examples of its application are multi-objective assignment problems (Laguna, Lourenco, & Marti, 2000) and linear ordering problems (LOPs) (Campos, Glover, Laguna, & Marti, 2001) among all. For further SS applications refer to a survey done by Glover et al. (2005).

2.3.2.5 Cuckoo Search

Developed by Yang and Deb (2009), Cuckoo Search (CS) is one of the recently-introduced metaheuristic algorithms. CS was inspired by the broom parasitism of some cuckoo species that lay their eggs in the nests of the birds of other species. In the basic version of CS, the cuckoos are illustrated as basic search agents. Eggs in nests serve as candidate solutions for the problem at hand and each egg is a metaphor for a new solution. The main goal is to use new and potentially better solution to replace a worse solution in the nest.

Based on the work done in (Yang & Deb, 2010), CS has shown itself as a very efficient algorithm for finding the global optima with high success rate. Yang (2010) showed that in some cases CS was superior to both PSO and GA in terms of efficiency and success rate. Moreover, primarily because of its effectiveness and simplicity, CS has managed to attract the attention of many researchers from different application fields and domain, refer to (Nguyen, Truong, & Phung, 2016; Teymourian, V.Kayvanfar, Komaki, & Zadeha, 2016; Sekhar & Mohanty, 2016; Elazim & Ali, 2016) for examples.

In terms of applications of CS to problems in the discrete domain, as of today there are not many cases. One of the first works that attempted to solve a discrete optimisation problem using CS was presented by Ouaarab et al. (2013). In their work, the authors used the basic and improved CSs to solve the travelling salesman problem. Further, CS has also found a recent and significant application to the NP-hard annual crop-planning problem by Chetty and Adewumi (2013).

More recently, Yang (2012) proposed a generalised version of CS called Flower Pollination Algorithm (FPA). The algorithm is based on the principle of pollination of flowers, however, it bears a lot of similarities with the CS. The major difference between these two is the application of the crossover operator in FPA.

2.3.3 Hybrid Algorithms

Nowadays, one of the biggest trends in the engineering optimisation is the hybridisation of metaheuristics. Hybrid algorithms exploit the complementary character of different optimisation strategies. As a result, choosing an adequate composition of several algorithmic concepts for hybridisation can be the key for achieving better performance in solving many hard optimisation problems. Several forms of hybridisation of metaheuristics exist. Talbi (2002) in his survey on hybrid metaheuristic proposed to classify hybridisation forms as follows:

- Hybridising metaheuristics with (meta-) heuristics
- Hybridising metaheuristics with constraint programming
- Hybridising metaheuristics with tree search techniques
- Hybridising metaheuristics with problem relaxation
- Hybridising metaheuristics with dynamic programming

The process of designing and implementing effective hybrid metaheuristic is rather complicated and requires broad knowledge of algorithmic techniques and specifics of the problem to which the algorithm is going to be applied. Nevertheless, despite such complexities, a large number of publications (Bluma, Puchingerb, Raidlc, & Roli, 2011) documents great success and benefits of various hybrid approaches.

In the review done by Kolisch and Hartmann (2006) on the latest state-of-the-art heuristics for the RCPSP, the best performance results were achieved indeed by a hybrid metaheuristic developed by Valls et al. (2003). This algorithm is based on the GA and introduces several changes to the original paradigm: a local improvement operator; a new selection mechanism; and new crossover operator specific for the RCPSP. The new crossover operator, called peak crossover, is designed to combine useful problem-specific information extracted from the parents for the purpose of generating high-quality offspring.

Moreover, in the last decade, more researchers resorted to the development of the hybrid approaches for the RCPSP. One of the first of such methodologies was introduced by Valls et al. (2003). The presented algorithm is a population-based method with a TS integrated into it that uses a topologically ordered random key (RK) as the representation of a solution. The solution neighbourhood is structured and explored by the application of three different types of local move operators. Likewise, Kochetov and Stolyar (2003) proposed an evolutionary algorithm that combines the GA, the path relinking and the tabu search.

In (2004), Valls et al. proposed another hybrid population-based methodology for the RCPSP. The authors applied a combination of scatter search (SS) and path relinking strategies. Moreover, for subsequent improvement of the results, the algorithm uses a forward and backwards schedule improvement technique, commonly referred to as double justification (Valls, Ballestín, & Quintanilla, 2005).

Tseng and Chen (2006) developed a hybrid metaheuristic which they applied to solve the RCPSP. The presented algorithm hybridises the ACO, GA and TS methods. In particular, ACO is used to create an initial population, GA to further improve it, and TS for supplementary modifications.

A more complicated way of a solution search space exploration was proposed by Debels et al. (2006). The developed algorithm represents a hybrid metaheuristic which is implemented by combining SS and the ideas of the electromagnetism. For this method, the authors developed a specialised representation of a solution and new intensification procedure.

Debels and Vanhoucke (2005) presented another hybrid methodology for the RCPSP, this time, based on the GA. The presented method deconstructs the RCPSP into various sub-problems which re further solved with the application of the GA. As the next step, the sub-solutions are assembled by the solution framework. In a similar fashion, Mendes et al. (2009) presented a GA that uses an RK as default representations of a solution.

Another hybrid methodology was presented by Agarwal et al. (2011). The authors proposed a *neurogenetic algorithm* for the RCPSP. The algorithm works by hybridising a population-based strategy that is based on a GA and local search algorithms and principles of the neural networks. Similarly to previously reviewed algorithms, the presented algorithm also operated on the AL representation.

Finally, an *improved immune algorithm* for the RCPSP was proposed by Wu et al. (2011). The presented approach is a population-based metaheuristic that emulates the immune system of living organisms. It uses an RK as solution representation scheme, random numbers of which are obtained by the chaotic generator, and incorporates a novel hypermutation mechanism.

2.3.4 Multimodal Optimisation Algorithms

One common characteristic of methodologies that were presented in this literature review so far is that they all focus on obtaining only one solution at the time.

Ikeda and Kobayashi (2000) examined fitness landscapes of the most popular combinatorial optimisation problems, such as JSSP and TSP, and demonstrated that these problems typically have *deceptive multimodal landscapes* (i.e. strong local optima exist far from the global optima). Moreover, the authors showed that the fitness landscapes of these problems have structures of a *big valley* (Boese, 1995), meaning that one area of the solution space might contain many solution candidates that tend to be very similar to each other. Depending on the problem instance, its fitness landscape might consist of several big valleys. If valley that contains the global optimum covers a much smaller part of the domain that the other valleys, this topology poses considerable challenges for search heuristics, as most searches are drawn toward the bottom of suboptimal valley. Ikeda and Kobayashi (2000) referred such topologies as UV-valleys, primarily for their structural appearance as can be seen Figure 2.1.

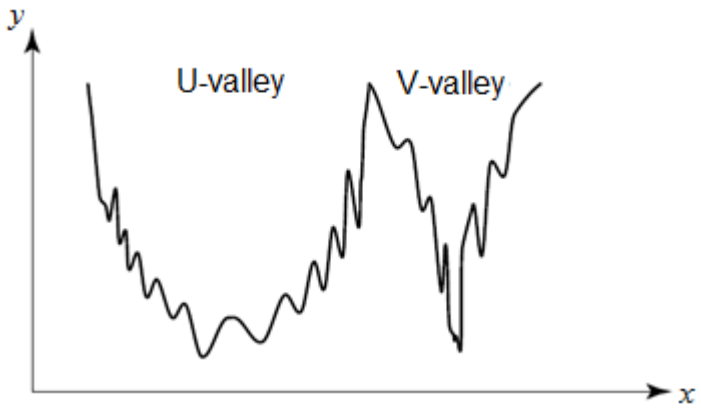

Figure 2.1 – Concept of U- and V-valleys

Czogalla and Fink (2009) also independently analysed a fitness landscape of the RCPSP and provided a statistical analysis of their research. The authors proved that the solution space of the RCPSP, similarly to other combinatorial optimisation problems, has a big valley structure and showed that good solutions tend to be close to other good solutions (but not too close) and they are spread all around the solution search space. The statistical analysis indicated that the landscape of the RCPSP consist of several interior plateau meaning that one instance of the problem can have multiple optimal solutions.

More recently, Pérez et al. (Pérez, Posada, & Lorenzana, 2015) applied Multi-Modal Genetic Algorithm (MMGA) to solve the Resource-Constrained Multi-Project Scheduling Problem (RCMPSP), which is the derivative of the standard RCPSP. In their work, the authors were able to prove that multiple optima can be obtained in the RCMPSP, as well as to demonstrate that multimodal techniques

provide better performance than other alternative commonly accepted methodologies for RCMPSP. Moreover, in his previous works (Pérez, Herrera, & Hernández, 2003; Pérez, Posada, & Herrera, 2012), Pérez successfully applied similar approach for solving the JSSP.

From the above-mentioned works, it can be concluded that globally multimodal landscapes of these problems consist of plural big valleys, and each of them has its own important local or global optima. In the example presented in Figure 2.1, a sample fitness landscape of a two-dimensional problem contains two big valleys. It is worth mentioning that U- and V-valleys are not explicitly identified by heuristics in the search process, as these are just relative concepts. From the presented example, it can be noted that one of the valleys seems to have a relatively better fitness than the other one at the beginning of the search process. If the global optima are located in a V-valley, traditional heuristics are likely to fail to find it, because the search will go toward a more promising area (U-valley). As the result, the effect of premature convergence will occur.

One of the possible ways of the diversification of a search process and elimination of possibilities of falling into local optima trap is to maintain the diversity of a population. This can be achieved by diverting the search process into various regions of the search space simultaneously. In the optimisation this is achieved by application of methods specifically designed for tackling multimodal optimisation problems. Every multimodal optimisation technique has to simultaneously fulfil two partially conflicting tasks: to locate multiple optima and to maintain a set of best solutions for diversity purpose.

Several attempts have been made to transform EC methods so that they could be applied in problems with globally multimodal landscapes (refer to (Eiben & Smith, 2003) for recent surveys). However, when tailoring such EC, there are a number of issues to be considered: 1) the division of a population into sub-populations; 2) preservation of these sub-populations; 3) connection of these populations to the existing optima in the solution space. In the literature, the most popular and effective technique that managed to address all three issues is niching.

In nature, an ecosystem consists of regions (niches). A niche represents a part of a habitat where a living thing makes its home. Each of these niches has a diverse set of characteristics and stimulates the formation and subsequent development of different types of species. Hence, species is a living thing that

lives in a niche. To allow the coexistence in their niches, the individuals that form a species share a set of similar biological features. Moreover, this set of biological species allows them to breed with each other, and, at the same time, makes the interbreeding among members of different species impossible. Usually, the individuals that form a species depend on the resources their niche provides to them. In the context of optimisation problems, each niche is represented by a region of a search space and is related to a peak of the fitness landscape. Species that inhabit that niche represent a solution candidates for the problem. In this respect, niching or speciation techniques have been proposed for the simultaneous evolution of subpopulations.

Mahfoud (1995) undertook a detailed comparison of popular sequential and parallel niching methods. By applying reviewed techniques on various multimodal problems, the author concluded that parallel niching outperforms sequential niching in all problems with intermediate to high complexity. Sequential niching was only able to solve problems of low complexity. The most competitive sequential niching techniques are those of, Beasley et al. (1993), and Li et al. (2004).

Many different parallel niching concepts and methodologies have been developed. The most prominent examples of such techniques are *species conservation* (SC) (Li, Balazs, & Parks, 2002), *clustering* (Yin & Germay, 1993), *fitness sharing* (FS) (Cavicchio, 1970), *crowding* (Jong, 1975), *clearing* (Petrowski, 1996), *restricted tournament selection* (RTS) (Harik, 1997), and *niching memetic algorithm* (NMA) (Moscato, 1989).

2.3.4.1 Species Conservation

Species Conservation (SC) is a recent technique that was introduced by Li et al. (2002) and it realises niching by utilising the idea of species. The technique is based on the concept of separation of the population into several species according to their similarity and the subsequent evolution of the species in parallel.

In SC, a species represents a subset of finite population P_N, which is composed of individuals that are considered to be similar enough to each other. The similarity is defined by calculating a distance d between individuals with respect to the species distance parameter σ_s. If a distance between two individuals is less

than half the species distance (i.e. $\sigma_s/2$), then they are assigned to the same species.

To divide a population into species and determine which individuals will be preserved into the next generation later, a set of species seeds X_s is established. In the beginning of the procedure, the population is sorted in descending order of fitness of its members. Then. After sorting is completed, the fittest member is then added to the X_s as the first species seed. Then, for each member of the population, a distance is calculated between him and species seeds in the X_s. If the distance is greater than $\sigma_s/2$, the individual is added to the X_s.

After all species have been established, the population is evolved by applying the usual genetic operators: selection, crossover, and mutation. Since some species may not survive the outcomes of these operations, they need to be copied into the next generation, thus prolonging their existence. To do so, the species seeds of the previous generation are replaced with the individuals of the same species in the next generation if their fitness is worse than of those in the previous generation. If no individuals of the same species are found in the next generation, the worst individual of the new population is replaced by the species seed. Since the species seeds are being taken from the previous generation, the number of species is always less than the population size.

Li et al. (2002) integrated SC technique into simple GA and presented Species Conserving Genetic Algorithm (SCGA). The performance of the SCGA was successfully tested on various multimodal test functions. The test results proved this technique to be very effective in locating multiple global optima. One of the biggest advantages of the SCGA is that it makes no distinction between genotypes and phenotypes. Therefore, the genetic operators are applied directly to individuals represented by arrays of real numbers, thus increasing the simplicity of the technique. The results of further experiments ran by Li and Wood (2009) confirmed the effectiveness and efficiency of this technique.

Parrot and Li (2004) and Li (2004) proposed a Species-based Particle Swarm Optimisation by applying SC concept to the basic PSO. The main strength of the basic PSO is the ability to adaptively adjust particles' positions based on the dynamic interactions with other particles in the population. Because of that, the algorithm becomes very well suited for application in problems with the multimodal landscape. Using the SC technique, the presented algorithms determine the neighbourhood best particles and use them to guide different

portions of the swarm population towards different optima. The proposed methodologies were successfully tested on various multimodal problems from the literature.

Analogously to the previous methodologies, Iwamatsu (2006) extended the original PSO with the SC technique. The algorithm works by dividing particle swarm into multiple species. Each species explores a different area of the search space independently by sharing information to its members. To provide information exchange between species, the immigration of particles from one species to another is implemented. The performance of the algorithm is compared with several other proposals on a set of multimodal test problems.

Ando et al. (2005) incorporated SC into a GA to receive Adaptive Isolation Model algorithm. The presented approach applies SC to detect clusters in the population which are identified as attractors in the fitness landscape. The subpopulations, which make-up clusters, are then isolated and each is optimised independently, whereas the regions of these subpopulations are suppressed. The purpose of isolation is to increase comprehensiveness, i.e. the probability of finding stronger attractors, and the overall efficiency of the multimodal process. In comparison to other SC-based methodologies, the algorithm does not require to configure species distance parameter σ_s, as it is estimated from the variance/covariance matrix of the subpopulations.

Stoean et al. (2010) presented a Topological Species Conservation Genetic Algorithm that integrates the conservation of the best successive local individuals with topological subpopulations separation, instead of the common radius-triggered manner. The algorithm inherits the ideas of SC of establishing and conserving dominating individuals, and, at the same time, uses the principles of the multinational *GA* to establish sub-populations and distinguish between basins of attraction. Such approach allows to control seed dynamics even further, both as replication and exploration are concerned, and eliminates the requirement of the species distance σ_s.

Dong et al. (2005) used SC technique in conjunction with mixed mutation strategy. The presented technique mixes Gaussian, Cauchy, Lévy, single-point and chaos mutations, which are then applied to the each individual in the population to generate an offspring according to a mixed strategy distribution. Mixed strategy distribution is dynamically adjusted based on the performance of

the mutation strategies. The addition of the SC eliminated the premature convergence.

Shibasaka et al. (2007) presented a Species-based Differential Evolution (SDE). The presented method is the first attempt to integrate SC into DE. The method showed promising results in locating multiple global optima.

2.3.4.2 Clustering

To promote the formation of the niches and at the same time eliminate the need for estimation of the niche radius parameter σ_{share} (needed in the majority of other niching techniques), Yin and Germay (1993) proposed clustering.

In the original clustering technique, niches are formed using the adaptive Macqueen's K-means clustering algorithm (MacQueen, 1967). The procedure of clustering (niches formation) is normally utilised in population-based metaheuristics and performed at each generation. The procedure begins with the initialisation of a fixed number (k) of seed points, referred to as the best k individuals. Using a minimum allowable distance d_{min} between niche centroids, the clusters are built around each seed point. Then, based on the values of d_{min} and d_{max} parameters, the remaining members of the population are added to these existing clusters or are used to form new ones. The final fitness of each individual is calculated using the following relation:

$$F_i = \frac{f_i}{n_c(1-(d_{i,c}/2d_{\max})^{\alpha})'} \tag{2}$$

Where n_c is the number of individuals in the niche containing the individual i and d_{max} is the maximum distance allowed between an individual and its niche centroid. As can be noted from (2), the estimation of a fitness is based on the distance $d_{i,c}$ between the individual i and its niche centroid. Such method of estimation significantly reduces the time complexity.

Yang et al. (2005) proposed Density Clustering technique. With the aim of preventing the loss of diversity, Yang et al. replaced the global selection procedure applied to the whole population with the local selection strategy which is applied to sub-populations instead. Thus, the species that are represented by sub-populations are dynamically identified using density-based clustering algorithm. The algorithm also includes a method for automatically calculating the clustering threshold.

Gan and Warwick (2001) proposed the Dynamic Niche Clustering. The developed technique represents an improvement to the original methodology and is a niching method that manages a separate population of overlapping fuzzy niches. Each of the fuzzy niches has independent radii which operate in the decoded parameter space and is maintained alongside the normal population. Moreover, the authors implemented a speedup process that is applied to the initial population with the goal of reducing the time complexity of the preliminary stages. It is demonstrated that the added process improves the overall robustness of the technique.

Streichert et al. (2004) proposed a variation of the clustering strategy for the EC. The basic idea of the proposal is to transfer the biological concept of non-interbreeding species living in separate ecological niches into EC. The technique artificially separates the initial population into species by locating the clusters of individuals in the search space, which naturally occur due to the general convergence of EC algorithms. These clusters are then separated into isolated subpopulations in which individuals compete and breed like in any traditional EC. Each of the subpopulations converges to a global/local optimum. The division of the population into species the diversity in the population.

Jelasity et al. (2001) proposed the Abstract Clustering technique for multimodal optimisation. To accelerate and parallelise existing search methods, the authors proposed to create clusters with the application of the hill climber technique. The communication between clusters is minimal and as the search goes on, the volume of clusters decreases. The process of clusters decrease is implemented in the similar fashion as cooling in the SA. The reason for limited communication between clusters is to ensure that each hill is explored only by one hill climber. The authors embedded their technique into a GA and the results of the evaluation confirmed its effectiveness.

Alamil et al. (2009) presented a fuzzy clustering-based PSO that does not require any prior information about the cluster radius σ_{share} and a number of known optima. The basic idea of this technique is to maintain and promote the formation of the parallel sub-swarms using fuzzy clustering. Since σ_{share} is dynamically adapted, a fine local tuning is used to improve the solution during the evolution of the process.

Another method based on fuzzy clustering is proposed by Alami et al. (2007). In the presented approach, namely Multi-Population Cultural Algorithm, with the

application of the fuzzy clustering technique, the entire population is divided into smaller subpopulations. These sub-populations and their belief spaces are kept isolated and handled by their own local cultural algorithm. Therefore, to follow the principle of social environment, the cultural exchange concept is implemented. Experimental results indicate that proposed methodology performs better than normal fitness sharing technique and demonstrate that the method is capable of optimising high dimensional multimodal functions.

Ling et al. (2008) utilised the clustering strategy into GA to eliminate the genetic drift that is introduced by the crowding strategy. To combine the clustering and crowding technique, the authors introduced a peak detection concept. The niches and clusters in the fitness landscape of a given problem are formed using the standard crowding strategy. Different niches can coexist in the same cluster and lead to the same optimal solutions. To remove the genetic drift caused by the tendency of crowding techniques to converge to numerous potential solutions simultaneously, the clustering operator is employed to stimulate exploration of the entire solution search space.

Passaro and Starita (2008) used k-means clustering technique in the conjunction with PSO for identifying niches within the swarm and locating multiple global optima.

2.3.4.3 Fitness Sharing

Amongst all proposed niching techniques, Fitness Sharing (FS) is the first one that attempted to deal directly with the locations and preservations of multiple solutions. The original concept was proposed by Goldberg (1987) and was further improved by Goldberg and Richardson (1987). The main idea of FS is to divide the population into different sub-groups according to the similarity of the individuals and within each of the subgroups, its members will share the relevant information among them.

FS works by modifying the search space and reducing the payoff in densely populated regions. As the outcome, the fitness of each individual within the subgroup is decreased by an amount nearly equal to the number of similar individuals. Typically, the shared fitness f_i' of an individual i with the fitness f_i is

$$f_i' = \frac{f_i}{m_i} \tag{3}$$

where m_i is a niche count parameter which represents the number of members with whom the individual shares his fitness. The niche count is the sum of the sharing function of all individuals in the population:

$$m_i = \sum_{j=1}^{N_P} sh(d_{ij}) \tag{4}$$

where N_P is the size of the population and $d_{i,j}$ is the distance between individuals i and j. Therefore, the sharing function (sh) estimates the level of similarity between members of the population and is calculated as follows:

$$sh(d_{ij}) = 1 - (\frac{d_{ij}}{\sigma_{share}})^{\alpha} \tag{5}$$

where σ_{share} represents the threshold of dissimilarity and α is the constant parameter which regulates the shape of the sharing function. If α is set to one, the resulting sharing function is the triangular sharing function (1987). The distance $d_{i,j}$ between two individuals is characterised by a similarity metric based on either genotypic or phenotypic similarity. Deb and Goldberg (1989) show that sharing based on phenotypic similarity may give slightly better results than sharing with genotypic similarity.

Goldberg and Wang (1998) proposed an alternative sharing scheme known as the Evolutionary Sharing. The scheme surpasses limitations of the original FS scheme by letting niches adapt to complex landscapes, thus, promoting a better distribution of solutions for problems with many poorly spaced optima. The presented technique is based on the principles of Monopolistic Competition in economics. In accordance with this principle, two populations are utilised – a population of customers and population of businessmen. The individuals from both populations attempt to maximise their interests by evolving nearby spaced niches consisting of the fittest individuals. The solutions to the problem at hand are represented by the members of businessman population.

Analogously, to overcome the limitations of the original technique, Della Cioppa et al. (2007) proposed Dynamic Fitness Sharing. The proposed method allows an explicit, dynamic identification of the species discovered at each generation. The authors implemented a species elitist strategy which consists of the localisation of species on the fitness landscape and application of the sharing mechanism to each of them. The performance of the method is assessed using

a set of standard multimodal test functions adopted from the literature. Experimental results confirm that the technique performs significantly better than original fitness sharing.

Horn (2002) proposed a composite of resource sharing and FS named Resource-Based FS. Unlike traditional FS techniques, the presented method exploits principles of the resource sharing which are based on nature's way to induce speciation during evolution. The explicit use of resources keeps the calculations of equilibrium points simple, whereas the path to the equilibrium does not lose key species along the way.

Thomsen (2004) integrated the FS concept with the DE to form the Sharing Differential Evolution. The sharing DE utilises the classical sharing technique described previously and uses the Euclidean distance for estimation of the similarity between individuals. At each iteration, the algorithm generates a number of offspring equal to the size of the parent population N_P. After N_P offspring is generated, the fitness of each individual is calculated using the sharing function and the worst half of the population is purged. The algorithm provides elitism by always preserving the individuals with the best un-scaled fitness.

Salazar-Lechuga and Rowe (2005) introduced the algorithm that combines the concepts of PSO and FS to tackle the multimodal optimisation problems. Ideas of the PSO are extended with the FS function, allowing the algorithm to spread particles along the Pareto front and guide the search in right directions. Because of that, the FS function is used in the objective space. This promotes the diversity in the population, as particles within highly populated areas in the objective space are less likely be followed. At each iteration, the particles with the highest fitness are inserted into the repository. The particles from the repository are then used to guide the search for the next generations and conserve a set of non-dominated solutions until the end of the run.

2.3.4.4 Crowding

Crowding (Jong, 1975) is motivated by the analogy of the competition for limited resources among individuals of the same species in the natural population. Its ideas are based on the principle that dissimilar individuals tend to reside in different niches and, as the result, they do not compete. In the original proposal, De Jong (1975) proposed a simple method which replaces individuals with lower

fitness with the newly generated individuals, assuming they are similar enough. The similarity between individuals, likewise in other niching techniques, is defined by a distance d between them (i.e. the closer individuals are, the more similar they are). In genotypic distance sharing the distance function is simply the Hamming distance, whereas in phenotypic distance sharing the distance function is defined using some problem-specific knowledge, the most common choice of which is the Euclidean distance.

The technique can be compared to a simple GA. The main difference between those two is that in crowding only a fraction of the global population (indicated *generation gap* G) reproduces and dies in each generation. As the result, in the crowding, new members of a particular species replace older members of that species. Such way the pre-existing diversity of the population is always maintained.

Moreover, in contrast to other niching methods, crowding does not assign individuals to fitness peak. Instead, the number of individuals that is assembled on the peak is largely determined by the size of that peak's basin of attraction. To estimate the size of a niche, a random sample of individuals is taken from the population, denoted by the crowding factor CF. If the value of CF is set too low, in some cases the individual from the population might be replaced by the new individual that is not similar enough, thus creating the replacement error. To overcome this problem, CF should be very large or equal to the number of individuals in the population. Because of the frequent occurrences of the replacement errors, the initial crowding of De Jong was shown to be of limited usefulness in multimodal optimisation (Deb & Goldberg, 1989).

Mahfoud (1992) reviewed De Jong's (1975) original crowding technique and proved its inability to maintain more than two peaks of a multimodal objective function, mainly due to the replacement errors that result from a genetic drift. As a solution to this issue, Mahfoud (1992) proposed the improvement of the original method called Deterministic Crowding. The objective of the deterministic crowding is to maintain diversity in the population, eliminate any parameters that require problem specific knowledge, reduce the occurrence of replacement errors, and improve selection mechanism. The proposed algorithm works by selecting two parents from the current population and randomly performing crossover and mutation on them. As the outcome of this procedure, two offspring are generated. Then the children replace the nearest parent if they have better

fitness. In the case of a tie, parents are preferred. The procedure is performed $N_p/2$ times (where N_p is the population size). Thus, deterministic crowding results in two sets of tournaments: (parent 1 against child 1 and parent 2 against child 2) or (parent 1 against child 2 and parent 2 against child 1). The set of tournaments that yields the closest competitions is held.

With the aim of improving the performance of GAs for multimodal optimisation in ill-scaled and locally multimodal domains, Ando et al. (2005) developed a Sample-Based Crowding. In the presented scheme, the pairs for tournament selection are determined based on a statistical comparison of their fitness values. The technique takes into account ranks of the parents among the sampled values in the selection process which are used to determine their indispensability. These measurements are scale-invariant, thus enabling the proposed method to search a domain without presuming a distance between optima and eliminating the need for scaling and correlating the variables.

Thomsen (2004) extended DE with a crowding technique to receive Crowding Differential Evolution. In crowding DE an offspring is generated by using the standard DE operators, which then competes against the most similar individuals in the current population. The individual is replaced if the value of his fitness is worse. To avoid a replacement error, the value of crowding factor CF is equal to the population size N_P.

Zaharie (2004) proposed a Multi-population Crowding Differential Evolution by integrating crowding technique into DE algorithm. Under this scheme, the initialisation of subpopulations is no longer necessary, as each subpopulation is now capable of locating multiple global optima. To avoid global processing, the use of crowding is only limited to the establishment of subpopulations.

Similarly, another integration of crowding into DE is presented by Kundu et al. (2013). To avoid the use of niching parameters that require prior knowledge about the fitness landscape, the authors used local mutation for search the solution space. Moreover, a speciation-based memory archive is integrated for regeneration of population after an environmental change is detected. The experimental analysis and comparison with other peer algorithms confirmed the effectiveness of the proposed method.

To handle multimodal optimisation problems, Angus (2009) incorporated the idea of crowding into ACO. The implementation of the crowding helps to maintain the diversity of the population, making the ACO more robust. During the

experimental evaluations, the algorithm was able to locate and maintain multiple spatially distributed near-optimal solutions for various multimodal test problems.

2.3.4.5 Clearing

Petrowski (1996) presented the clearing procedure. It draws inspiration from the principle of sharing of limited resources among the strongest members of the niche and elimination of weaker individuals of the same niche.

Typically integrated into GAs, the clearing procedure is applied between the processes of the fitness evaluation of all members of population and selection for the crossover. Similarly to other niching methods, the clearing uses a dissimilarity measure between individuals to determine whether they belong to the same subpopulation or not. This value could be the Hamming distance for binary coded genotypes or the Euclidian distance for real-coded genotypes. Each subpopulation contains a dominant individual: the one that has the best fitness. If the individual belongs to a given subpopulation, then its dissimilarity with the dominant is less than a given threshold σ_c (*clearing radius*).

In this method, each subpopulation contains a dominant individual: the one with the best fitness. The fact that an individual belongs to a subpopulation means that it is at less than a threshold σ_c from this subpopulation's dominant. But, differently from sharing, in clearing the dominant's fitness is preserved and all the other individuals have their finesses zeroed (in the case of maximisation problem). In other words, all resources of the niche are given to only one individual: the so-called *winner*.

Petrowski (1996) generalised the basic clearing technique, stating that each niche can by dominated by more than one winner. The maximum amount of winners that can dominate the niche is defined by the *capacity* (k) parameter. The niche's capacity can range from 1 to the population size, therefore, the niching effect can be, respectively, maximised or minimised as convenient.

Dick (2010) developed an extension to the original clearing method called *local clearing.* The presented technique uses information gained during the course of evolution to accurately determine the correct niche radius in both real-parameter and discrete optimisation problems. This adaptability of the niche radius is based on the fact that the parallel instances of niching within local clearing allow each subpopulation to focus on different, yet partially overlapping, subsets of optima. When these subsets are combined, the system gets a clearer picture of the

location of optima within the total fitness landscape and subsequently can more accurately predict the correct niche radius.

Variation of the clearing called Context-Based Clearing is presented by Fayek et al. (2010). The presented approach is a clearing procedure that makes use of a context information with the aim of preventing the elimination of candidates that may lead to significant optima. In the case of the technique, context refers to the fitness distribution within a certain area around pivot elements. Within the same area, if the candidate has similar fitness, it is assumed that all candidates converge to the same optima; hence, the whole area can be cleared. However, if candidates' finesses differ significantly, clearing the whole set may cause a loss of important date. The procedure performs clearing according to the heterogeneity of the individuals within the subpopulation, whereas heterogeneity is measured using the standard deviation of individuals' fitness.

As a solution to the clearing's main weakness – the estimation of the clearing radius – Sacco et al. (2004) proposed Fuzzy Clearing. While in the standard clearing a dominating individual dominates those that are within his clearing radius, in the proposed technique the population is divided into clusters, hence the use of radius parameter is no longer needed. To cluster the population, the authors proposed a fuzzy class separation algorithm. The algorithm borrows from a fuzzy logic a concept of pertinence that denotes a degree of association of an individuals to a given class. The proposed technique is successfully tested on a set of multimodal problems.

Qu et al. (2012) embedded clearing into DE algorithm. In the presented method, the initial population is divided into three equal subpopulations. The value of clearing radius for each subpopulation is different and is related to the problem's search range. During the selection phase, subpopulations exchange with the relevant information.

In (2014), Sacco et al. proposed a clearing paradigm that is based on the works of Sacco et al. (2004) and Qu et al. (2012). The technique uses a clustering heuristic based on the topographical information on the objective function and new mutation operator, taken from the DE (Qu, Liang, Suganthan, & Chen, 2012). The presented clearing variant, namely topographical clearing, was applied to DE algorithm, however, as the authors state, it can be applied to any evolutionary or swarm-based technique.

2.3.4.6 Restricted Tournament Selection

Restricted Tournament Selection (RTS), introduced by Harik (1997), is a modified tournament selection for multimodal optimisation and its main idea is to allow the GAs to choose which individuals will be replaced by a new pair of individuals.

As in deterministic crowding, RTS randomly selects two parents from the population and creates two offspring by applying crossover and mutation operators. For each generated offspring, the algorithm randomly selects sample individuals from the population, the size of which is denoted by w (windows size, analogous to CF in crowding), and find the nearest one to the offspring, by applying the similarity distance measure. The distance can either be Euclidean (for real values) or Hamming (for binary-coded variables). The closest individual within the w sample competes with the offspring to determine the one with better fitness. If the offspring has higher fitness, the opponent is replaced. Such type of tournament restricts members of the population from competing with others that are not similar enough.

Harik (1997) tests his model on several multimodal real-world problems with the number of peaks varying from 5 to 32. The algorithm proved to be capable of maintaining individuals at all peaks, even though some peaks increasingly lost an amount of individuals. Moreover, the algorithm managed maintained all global optima in all multimodal test problems.

Roy and Parmee (1996) presented an Adaptive Restricted Tournament Selection integrated into a GA for tackling multimodal optimisation problems. The main difference between the standard RTS and its adaptive variant is that the former requires no prior knowledge about the distribution of the optima on the fitness landscape to distribute the final population on different peaks.

Qu and Suganthan (2010) integrated the concept of the RTS into DE algorithm. The developed algorithm works by maintaining two different populations in parallel, where the size of each population is denoted by a windows size w. Each population generates a set of offsprings which then compete with members of both populations. Each offspring is compared against the closest to him a member of the population (based on the Euclidean distance). If the offspring has the higher fitness than the individual competing against him, offspring replaces this individual.

2.3.4.7 Niching Memetic Algorithm

Niching Memetic Algorithm (NMA), first introduced by Moscato (1989), represents an extension of the sequential niching technique of Beasley et al. (1993) proposed for application in the multimodal optimisation problems. The algorithm incorporates a gradient-based local search process that makes use of a derating function along with niching and clearing techniques. To promote the exploration of previously unvisited regions of the search space, the added process is used to penalise the individuals that are residing in regions which contain the already located optima. Similarly to other niching techniques, the NMA requires the use of a niche radius. However, the performance of the algorithm is not highly sensitive to the value of this parameter. In problems where the number and distribution of the optima are unknown, this can be considered as an advantage.

The process of NMA operation begins with the initialisation of the population of randomly or heuristically created individuals. After that, two additional parameters need to be configured: the *total number of optimal solutions* J_{Total} and *niche radius* σ_c. Then, at each iteration, a new generation is obtained by applying the usual genetic operators (i.e. evaluation, selection, crossover, and mutation). In each generation, individuals in the population move toward the nearest peak following a hill-climbing gradient-based algorithm. If at some point during this process an individual leaves the pre-specified region of the search space, the corresponding variables take the boundary value assigned. If J optimal solutions have been already located (with $J < J_{Total}$), the distances from each individual in the population to their nearest optimal solutions are determined. These distances together with the niche radius σ_c are used to assign an effective fitness function to each individual in the population. Therefore, the closer the individuals are to previously located optima, the lower his effective fitness is. Once the individuals in the population have been ordered in accordance to their effective fitness, the selection begins. Individuals with the high effective fitness value have an advantage over others individuals in the form of larger survival probabilities, which are assigned following the order position. Because of that, the individuals that lie within a niche radius of located optima are eliminated, thus promoting the occupation of yet unvisited niches.

The performance of NMA is not highly sensitive to the choice of the σ_c parameter. Moreover, in comparison to other niching methods, NMA does not

need to maintain permanent subpopulations around each found optima, as it only requires to store locations of the found peaks.

Vitela and Castano (2008) extended NMA to propose Sequential Niching Memetic Algorithm. The authors incorporated a Gaussian derating function with clearing in a real-coded memeting algorithm into a single point local search technique to accurately locate all optima (both local and global) in pre-specified regions of the solution space. Moreover, at each iteration, a local improvement algorithm as applied to every member of the population, substituting the original population members with the resulting solutions. The proposed algorithm uses the parent-centric real-parameter crossover operator which together with exploration and intensification phases efficiently searches the solution space. Performance measurements with test functions used by other authors show a high level of success in locating all optima and outperforming several other methodologies.

Sheng et al. (2008) integrated NMA into a GA. The authors suggested a unified criterion for simultaneous clustering and feature selection based on a scatter separability index, which is then optimised by the proposed algorithm. In order to allow simultaneous clustering and feature selection without the number of the cluster being known a priori, a composite representation is devised to encode both feature selection and cluster centres with a variable number of clusters. As a consequence, the crossover and mutation operators are suitably modified to tackle the concept of composite chromosomes with variable length. Additionally, the authors hybridised the proposed procedure with additional local search operators, where are introduced to refine the feature selection and clusters centres. These local searches move solutions toward local optima and allow a significant improvement in the computational efficiency. Finally, a niching method is integrated with the resulting hybrid GA to preserve the population diversity and prevent premature convergence.

2.4 Summary

Various strategies and methodologies have been proposed for dealing with standard deterministic RCPSPs, starting with simple exact methods, like a branch and bound algorithm (Brucker, Knust, Schoo, & Thiele, 1998), and ending with more advanced hybrid heuristics, like neurogenetic algorithm (Agarwal, Colak, &

Erenguc, 2011) and chaos-based improved immune algorithm (Wu, Wan, Shukla, & Li, 2011). Despite the variety of the proposed methodologies, the best performance results were achieved by algorithms belonging to a class of hybrid metaheuristics. In particular, in the survey on latest state-of-the-art methodologies for the RCPSP (Kolisch & Hartmann, 2006), the algorithm with the best performance is hybrid GA (Valls, Ballestin, & Quintanilla, 2003). In more recent years, more competitive heuristics for the RCPSP have been proposed. The most prominent examples of those are SS (Paraskevopoulos, Tarantilis, & Ioannou, 2012) and GA (Zhu, Li, & Shen, 2011).

In comparison to the deterministic RCPSP, the amount of methodologies proposed for solving SRCPSP is significantly lower. In the literature, there are two types of strategies recognised: use of scheduling policies, and use of heuristics. Scheduling policies are computationally fast and relatively easy to implement, however, they are only effective for small scale SRCPSP instances. Therefore, in this regard, application of the heuristic methods remains as the only reasonable approach. Out of all found heuristics for the SRCPSP, the best performance results were achieved by a two-phase GA developed by Ashtiani et al. (2011). However, the main drawback of this method is its static nature. To contradict this issue, several DP methodologies have been proposed for the RCPSP (Choi, Realff, & Lee, 2004; Haitao & Womer, 2015), however, their performance evaluation did not show any significant improvements.

The majority of algorithms for both of these problems either operate on the AL or RK representations or use customised adaptations of these two. For local search, were incorporated such mechanisms as double justification (Valls, Ballestín, & Quintanilla, 2005), local moves and activity swaps (Bouleimen & Lecocq, 2003). For crossover, the most commonly used are single-point or two-point crossover operators (Hartmann, 1998).

Nevertheless, despite the variety of the proposed methodologies, all of them deal with locating only one global optimum. The statistical analysis of the RCPSP fitness landscape, done by Czogalla and Fink (2009), indicated that the landscape of this problem is filled with a relatively high amount of interior plateau, meaning that for one problem there might be several global solutions. In the optimisation point of view, it is highly desirable to locate multiple optima: as it helps to maintain a diversity in the population, thus reducing chances of falling

into local optima trap and increasing chances of finding a global solution; and can provide an alternative, potentially better and more innovative, outcome result.

Chapter 3 Optimisation Model

The problem considered in this thesis, namely HARNet project management problem (HPMP), can be characterised as a special case of the RCPSP and it represents an optimisation model for scheduling projects in uncertain environment.

The chapter is split into five parts. The first part outlines the need for a new optimisation model.

In the second part, basic mathematical definitions of the proposed model are provided. These are standard definitions that coincide with definitions of the standard deterministic RCPSP.

The third section of this chapter gives a formal explanation of the nature of variability of activity durations and shows how it can be influenced by resources through them gaining experience.

The fourth section of the chapter outlines two objectives (primary and secondary) that are to be optimised.

Lastly, the overall problem statement is presented, which includes explanation of how the solution is obtained and what type of methodology needs to be applied to find it.

3.1 Problem Statement

Even though SRCPSP has received some portions of attention in the literature (Herroelen & Leus, 2005), in many cases the analysis of sources and causes of possible variabilities and their relation to the stochastic nature of activity durations has been avoided. Because of that, in the majority of publications on the SRCPSP, the stochastic durations of all activities follow predefined distributions. Examples of applied distributions are the triangular (Cho & Eppinger, 2005), uniform (2009), exponential (Vonder, Demeulemeester, & Herroelen, 2007), beta (Lamas & Demeulemeester, 2015), and normal (Bui, Michalewicz, Parkinson, & Abello, 2012).

In the real world, however, stochastic nature of durations can be reliant on many factors, such as the time period when activity is executed and efficiency of the allocated resource. For instance, quite often organisations have to initiate

new and arduous projects that would require the collaboration of many technical and managerial groups of people and, in some cases, some of the groups that participate in the project execution might have no relevant experience of working. However, as the execution of the project progresses, by participating in the completion of some of the project's activities, the maturity and efficiency of these groups are improving. As the result, the groups' capabilities to execute successive activities are expected to improve as well. Therefore, under such circumstances, the mean duration of an activity will reduce if it is started later in time and is allocated more efficient resources. In project scheduling with uncertainties, there are cases which also consider other external factors of influence on the activity duration (Davenport & Beck, 2001), however, for a long-term period planning these factors are very hard to predict and, thus, may create a high level of duration variability. Because of that, optimisation model proposed in this chapter only considers two things: period of time when activity is executed and experience of the allocated resource.

Various proposals have been made to address the above-described issues. Drezet and Billaut (2008) studied a variant of RCPSP that considers allocation of labour resources to the activities with varying in time resource requirements. The authors provide an integer linear formulation of the problem and use greedy algorithm to solve it. Talbot (1982) surveyed generalised version of the RCPSP in which activity durations and resources can be balanced by each other. Such variant of RCPSP can also be regarded as a special case of the multimode RCPSP (Mori & Tseng, 1997), in which modes represent different combinations of activity durations and resource requirements. Golenko-Ginzburg and Gonik (1997) were one of the first who studied the issue of duration/resource trade-off in the SRCPSP. In the problem described by the authors, activities have stochastic durations, nature of which is linearly dependent on the number of allocated resources.

Nevertheless, the common trait of the above-mentioned proposals is that for the duration/resource trade-off only the only consider amount of allocated resources. Other factors that can influence the duration of activities, such as efficiencies of resources, are not taken into account. To reflect this issue, Xiong et al. (2016) proposed an alternative probability distribution model that considers a case where durations of activities depend on the efficiency of the distributed resource, its ability to learn, and execution environment. In the context of the

proposed probability distribution model, efficiency of a resource influences the speed at which an activity is implemented, whereas the learnability reflects the rate at which the efficiency is gained. Unlike the traditional SRCPSPs, where stochastic durations are modelled using deterministic distributions, to incorporate the above-described effects, (Xiong, Leusb, Yanga, & Abbass, 2016) propose to use the mean of the durations that may decrease as the resource efficiency has improved via experience. The mean of the durations, in this case, reflects the overall capability to execute an activity. Such classification of the stochastic distribution has allowed the authors to model the resource efficiency and level of uncertainty as functions of time, which may affect the durations of activities, making them time- and resource-dependent. Therefore, following the above-described concept, it is now possible to create a special version of the RCPSP which will take into account the variable nature of activity durations.

3.2 Basic Definitions

Basic definitions of the proposed optimisation model, named HARNet project management problem (HPMP), coincide with those that formalise the deterministic RCPSP. Following the same mathematical definitions as defined in Chapter 2, a project is denoted by $\boldsymbol{V} = \{0, 1, \ldots, n, n+1\}$, consisting of 1 to n non-dummy (active) activities and activities 0 and $n+1$ are dummy activities. Activity processing times are defined by a set $\boldsymbol{p} = \{p_0, p_1, \ldots, p_n, p_{n+1}\}$, whereas activity starting and completion times are denoted by $\boldsymbol{S} = \{S_0, S_1, \ldots, S_n, S_{n+1}\}$ and $\boldsymbol{C} = \{C_0, C_1, \ldots, C_n, C_{n+1}\}$, respectively. If no volatilities are assumed, deterministic duration and starting time of an activity j are respectively denoted by p_j and S_j. Each of the project's activities requires resources for its execution. Resources are represented by a set $\boldsymbol{\mathcal{R}}^{\rho}$, which consists of k resource types. During its execution, for each period of duration t, activity j requires r_{jk}^{ρ} units of resource k, total availability of which is R_k^{ρ}. Since resources are renewable, after the execution of an activity is completed, the capacities of previously used resources are restored. Limited resource capacities imply that during certain periods of time several activities may require the same type of resource, which, due to its limited availability, can only be used by one of them at a time. A set of active activities, denoted by $\boldsymbol{V}_t$, during period t in the schedule $\boldsymbol{S}$ can be formalised as follows:

$$\boldsymbol{V}_t = \{\, j \in \boldsymbol{V} \mid S_j \le \mathrm{t} < C_j \} \quad j = 1, \ldots n \tag{6}$$

Following the above definitions, the resource constraints of resource type ***k*** are expressed as follows:

$$\sum_{j \in V_t} r_{jk}^{\rho} \le R_k^{\rho} \quad \forall R_k^{\rho} \in \Re \quad \forall t \ge 0 \tag{7}$$

where both r_{jk}^{ρ} and R_k^{ρ} are integers which represent resource requirement and resource capacity, respectively.

For their execution, activities may require different types of resources, examples of which can include manpower, specialised equipment or premises.

Some of the activities in the project might be inter-related with each other. In project scheduling, the most widely considered kind of relationships between activities is “end-to-start” relation. Such relationship implies that execution of activity can begin only after execution of all of its predecessors has been completed. The precedence relationship is characterised by a binary relation $\boldsymbol{E}$ and is presumed to be irreflexive and transitive. Notation $(i, j) \in \boldsymbol{E}$ means that activity j can only be started once the i is completed. Notation $(0, i) \in \boldsymbol{E}$ with $i>0$ means that dummy activity 0 is predecessor of all project activities. Analogously, notation $(i, n+1) \in \boldsymbol{E}$, with $i<n+1$ means that $n+1$ is successor of all project activities. A precedence graph (commonly known as project network) $G = \{\boldsymbol{V}, \boldsymbol{E}\}$ is inferred. The nodes in the graph correspond to activities and arcs that are connecting the nodes correspond to precedence relationships between them. Following the above definitions, the precedence constraints are expressed as follows:

$$S_j - S_i \ge p_i \quad \forall (i, j) \in E \tag{8}$$

3.3 Variable Activity Durations

Following the concept proposed in (Xiong, Leusb, Yanga, & Abbass, 2016), the activity durations for the proposed optimisation model can be described mathematically as follows. An activity i has a duration p_i and for its execution it requires resources k. HPMP differentiates between two types of resources:

- Improvable resources - the ones that can gain experience and improve over time (e.g. manpower)

- Invariant resources – the ones that have constant efficiency factors (e.g. specialised equipment)

The main difference between the improvable and invariant resources is the ability of the former ones to influence the duration of activities to which they are currently assigned through the experience that they have gained from the execution of previous activities. To accommodate the effect of experience gain, in addition to resource capacities, improvable resources have two additional parameters:

- Efficiency coefficient e_k - represents the maximum efficiency gain that can be achieved by unexperienced resource
- Learning coefficient l_k - quantifies how long it will take for a resource to achieve its maximum efficiency through learning

where $e_k \in [0, 1]$. Value $e_k = 0.25$, for instance, would mean that a highly experienced resource R_k can be up to 25% more efficient than a starter, whereas a higher value for l_k would indicate more time is needed for resource k to reach the maximal potential. In the context of the proposed optimisation model, efficiency of a resource is the amount of experience that it currently has that can be used to reduce the duration of an activity. At the start of the project, when resources have no experience and their efficiencies are at the lowest levels, the duration of an activity i is p_i^*. As resources gain more experience, durations of activities to which they are assigned reduce. Therefore, the duration of an activity i at given moment of time $p_i(t)$ can be defined as follows:

$$p_i(t) = \frac{p_i^*}{g_i(t)} \tag{9}$$

where t is the starting time of an activity i and $g_i(t)$ is the function for estimating resource efficiency or operating speed (Golenko-Ginzburg & Gonik, 1997). It is worth noting, that $g_i(t)$ is a non-decreasing function of t that depends on the resources allocated for activity i.

Efficiency of each resource type k is represented as $E_k(t)$. To represent the time that resource R_k has been working until the current time instance t, the notation $W_k(t)$ is used which is calculated as:

$$W_k(t) = \frac{\sum_{m=0}^{t} r_{km}^{\rho}}{R_k^{\rho}} \tag{10}$$

where r_{km}^{ρ} is the availability of resource k at given moment of time m with total capacity R_k^{ρ}.

The relationship between $E_k(t)$ and $W_k(t)$ is a non-increasing function that relies on the specifics of the project's setting and characteristics. Moreover, it has to obey the following rule: when $W_k(t)=0$, the experience of resource k is at the lowest level, hence it does not contribute to any reduction of the activity duration. Thus, the relationship $E_k(t)$ and $W_k(t)$ relationship is formalised as follows:

$$E_k(t) = \begin{cases} 1 + \dfrac{e_k}{\exp(\dfrac{l_k}{W_k(t)})}, & W_k(t) > 0 \\ 1, & W_k(t) = 0 \end{cases} \tag{11}$$

where e_k and l_k are efficiency and learning coefficients of a resource R_k, respectively. Further, from (11) it can be also noted that the value of $E_k(t)$ is always in the range of $[1, 1 + e_k]$.

In the real-life, the value of parameters e_k and l_k can be estimated through a managerial statement that will be based on relevant experience and knowledge gained through working with these resources (i.e. people) on previous projects. For example, a managerial statement like "100% of the maximum learning effect for resource type k can be achieved after 25 weeks of work" can be be mathematically interpreted as 100% = $exp(-l_k / 25)$, which yields that $l_k = 25 * (-ln(1.0))$.

Given that activity i is going to be started at time t, its overall efficiency can be estimated as the average efficiency of all required types of resources over required resource units and is defined as follows:

$$\overline{E^i(t)} = \sum_{k=1}^{R} \frac{E_k(t)}{r_{ik}^{\rho}} \tag{12}$$

Following the above definition, the function for operating speed estimation is now defined as

$$g_i(t) = \begin{cases} \overline{E^i(t)} & t > 0 \\ 1 & t = 0 \end{cases} \tag{13}$$

Function $g_i(t)$ is time-based and depends on the decision variables for resource allocation at given instant of time t. The received value can then be used in (9)

for estimating how the duration activity i is going to be affected if it is scheduled to be executed at time t.

The min and max processing times (i.e. durations) of activity i, denoted respectively by p_i^{min} and p_i^{max}, are calculated as:

$$p_i^{\min} = (1 - \overline{e_k})\, p_i * \qquad p_i^{\max} = p_i * \tag{14}$$

3.4 Optimisation Objectives

In practice, successful completion of a project may be subject to optimisation of several objectives. In the context of optimisation, these objectives are used to assess quality of the schedules that are produced to plan the execution of the project. The optimisation model proposed in this thesis considers optimisation of two objectives: makespan minimisation (primary objective) and resource efficiency balancing (secondary objective).

3.4.1 Makespan Minimisation

The main objective of the majority of problems that are derivatives of the RCPSP is the minimisation of the project's duration (i.e. the makespan). Similarly, for problems which consider project scheduling with variable activity durations the minimal makespan is also used as a primary measure to assess a quality of the produced schedule. Analytical evaluation of the expected project duration is typically highly intractable and is usually estimated by means of simulations. Therefore, the first objective is to minimise f_1, which is defined as follows:

$$f_1 = \frac{\sum_{p=1}^{P} C^{p}_{n+1}}{P} \tag{15}$$

C^{P}_{n+1} is the completion time of the last dummy activity achieved in the p-th simulation replication. In project scheduling, the number of simulations P typically is in the ranges between 25 (Stork, 2001) and 1000 (Ballestin & Leus, 2009).

3.4.2 Resource Efficiency Balancing

For successful completion of complex projects, stable and robust groups of staff (i.e. resources) are required to tackle future complex activities. If some members of staff of the project team have very high efficiency, while others lack experience, then the ability of the project team to perform activities efficiently is under risk. If,

for example, activity requires participation of two groups of staff out of the project team, then the group of more experienced personnel would need to wait for another group to finish their part. In an ideal scenario, the competency and maturity of all staff needs to be at similar levels.

In order to assess stability and robustness of the produced schedule, a variance index is used, which has been successfully utilised in other scheduling and planning problems, such as military capability planning problem (Abbass, et al., 2008). Therefore, the second objective is to minimise f_2 (i.e. estimation the balance of resource efficiency) which is defined as follows:

$$f_2 = \frac{\sum_{k=1}^{R} \sigma_k}{\overline{\sigma}} \tag{16}$$

where σ_k is the standard deviation of the efficiency of resource type R_k and $\overline{\sigma}$ is mean of standard deviations. σ_k can estimated as follows:

$$\sigma_k = \sqrt{\frac{1}{R_k}(E_k - \overline{E})^2} \tag{17}$$

where $\overline{E}$ is the average efficiency of all resource types. Lower values of f_2 signify better balance of resource efficiencies among different types of resources. The $f_2 = 0$ corresponds to a perfect balance, meaning that all resources have equal efficiency.

In the nutshell, (16) represents a fairness measure which ensures that experience gain between all resources is on equal level. Another way of measuring fairness would be via *chi-squared test* (Greenwood & Nikulin, 1996) which examines the differences with categorical variables and compares the actual observations with expectations

3.5 Optimisation Problem

Herein, for optimisation of the objectives, the problem exploits multimodal properties of the RCPSP. First, the problem is solved via optimisation of the objective f_1 by finding a set of best solutions with minimal makespan. Then, for each of the solutions in this set, the second objective f_2 is calculated.

Therefore, the best way of approaching this problem is via application of a metaheuristic algorithm specifically designed for tackling multimodal optimisation

problems and that is capable of obtaining multiple global solutions. For the reasons of having multiple objectives (one primary and one secondary), first, the algorithm will obtain a set of best schedules with minimal makespan, thus completing objective f_1. Then, out of this set, the most efficient schedule is going to be chosen, satisfying objective f_2.

Following the above definition, the problem can be formalised as follows:

$$\textit{Min:} \quad \frac{\sum_{p=1}^{P} C_{n+1}^{p}}{P} \tag{18}$$

$$\frac{\sum_{k=1}^{R} \sigma_k}{\overline{\sigma}} \tag{19}$$

$$\textit{Subject to:} \quad \sum_{j \in V_t} r_{jk}^{\rho} \leq R_k^{\rho} \quad \forall R_k^{\rho} \in \Re \quad \forall t \geq 0 \tag{20}$$

$$S_j - S_i \geq p_i \quad \forall (i, j) \in E \tag{21}$$

where V_t is as defined in section 3.2.

The presented problem is a variation of the RCPSP and can be regarded as a NP-hard combinatorial optimisation problem (Blazewicz, Lenstra, & Kan, 1983) with a complex multimodal fitness landscape (Czogalla & Fink, 2009).

3.6 Summary

Optimisation model, namely HPMP, presented in this chapter represents a special case of the RCPSP in which activity durations follow probability distribution model that is dependent on resource efficiency, experience, and learnability of resources, where resource in the HPMP case refers to human resource, i.e. members of the project team. Resource efficiency reflects the speed at which an activity can be implemented by the project team; experience is the total amount of time that members of the project team have previously spent on working on a similar problem; and learnability is the reflection of how quickly resource acquires its experience. As the result, the duration of an activity may be shortened with the increase of resource efficiency.

The HPMP considers optimisation of two objectives: minimisation of makespan (primary) and balance of resource efficiency (secondary).

Optimisation of these objectives exploited multimodal properties of RCPSPs and is achieved via application of metaheuristic algorithm tailored specifically for multimodal optimisation problems and that is capable of obtaining multiple solution candidates. First, the algorithm obtains set of best solutions with minimal makespan. Then, from this set, the algorithm selects the most efficiently balanced one.

Being variation of the RCPSP, the problem is NP-hard combinatorial optimisation problem with a complex multimodal fitness landscape. Therefore, the application of metaheuristic algorithms for multimodal optimisation problems is justified.

Chapter 4 Methodologies

This chapter presents four new algorithms developed during the course of this PhD study, namely

- Discrete Cuckoo Search (DCS) algorithm
- Discrete Flower Pollination Algorithm (DFPA)
- Improved Discrete Cuckoo Search (IDCS) algorithm
- Discrete Species Conserving Cuckoo Search (DSCCS) algorithm

The Discrete Cuckoo Search (DCS) and Discrete Flower Pollination Algorithm (DFPA) are population-based metaheuristic algorithms adapted from the Cuckoo Search (CS) (Yang & Deb, 2009) and Flower Pollination Algorithm (FPA) (Yang X.-S. , 2012). Previously, in the majority of cases CS and FPA had only been applied to problems in the continuous domain and demonstrated to be very effective in finding global optima with high success rate and, in some cases, even managed to outperform such popular metaheuristics as Genetic Algorithm (GA) and Particle Swarm Optimisation (PSO) in terms of efficiency and success rate. Nevertheless, at the time of writing this thesis, CS and FPA had only limited number of applications to optimisation problems in the discrete domain (Yang X. , 2010). The most prominent examples are the travelling salesman problem (TSP) (Ouaarab, Ahiod, & Yang, 2013) and the annual crop-planning problem (Chetty & Adewumi, 2013). In both of these examples, the algorithms showed competitve levels of performance, which validated their applicability for optimation problems in the discrete domain.

Performance of DCS and DFPA is evaluted using benchmark instances from Project Scheduling Problems Library (PSPLIB) (Kolisch & Sprecher, 1997). The results of evaluation can be regarded as satisfactory as the algorithms were able to outperform such heuristics as GA and Simulated Annealing (SA). Nevertheless, their performance can be furtherly improved by addressing some of the limitations that these algorithms have: inefficienct solutions representaions scheme and use of context-free operators. To address these limitations, the Improved Discrete Cuckoo Search (IDCS) is introduced in the next section. IDCS introduces several changes to the original DCS paradigm:

- addition of a new mechanic aimed at improving the quality of received results but with less iterations;

- new solution representation scheme specific for RCPSP and its stochastic variant; and
- novel local search and crossover operators, based on the newly-introduced solution representation scheme.

Similarly to DCS and DFPA, performance of IDCS is testing using benchmark instances from PSPLIB. This time, the results of performance evaluation are compared against state-of-the-art heuristics for RCPSP where IDCS was able to appear in top ranks. Nevertheless, one of the limitations of IDCS is the inability to obtain multiple solutions candidates at once, hence Discrete Species Conserving Cuckoo Search (DSCCS) is introduced in the next section.

DSCCS is the result of integration of the Species Conservation (SC) (Li, Balazs, & Parks, 2002) technique into IDCS. SC technique is a method of evolving parallel sub-populations integration of which allows the algorithm to obtain multiple global solutions. The technique is based on distributed elitism, achieved by identifying in each generation a set of prime individuals that are considered to be worth preserving into the next generation. The formation of species allows to divide the search space into smaller regions, making each species focused on searching for solutions within the specified region. This creates an opportunity for a finer search for a local best optimum, provides higher chances of finding global optima, as well as enables the algorithm to obtain multiple solution candidates, thus making it suitable for applications in multimodal scenarios.

4.1 Discrete Cuckoo Search

Out of all reviewed metaheuristics during the literature review, the one that has not been applied, as of yet, to the RCPSP is the CS. In the previous works of Yang and Deb (2010) CS has demonstrated to be a very efficient algorithm for solving continuous optimisation problems and in some cases it was shown to be superior to both PSO and GA in terms of efficiency and success rate. However, as of today, CS has been primarily used in optimisation of problems with continuous domain (Nguyen, Truong, & Phung, 2016; Teymourian, V.Kayvanfar, Komaki, & Zadeha, 2016; Sekhar & Mohanty, 2016). One of the first attempts to apply CS to solve a problem with a discrete domain was done by Ouarrab et al. (2013). In their work, the authors applied CS to solve the TSP.

4.1.1 Cuckoo Search

CS is a metaheuristic search algorithm, which has been recently proposed by Yang and Deb (2009). The ideas of the algorithm take inspiration from the reproduction strategy of some cuckoo species that lay their eggs in the nests of other host birds of different species. The host birds in their turn may discover that the eggs are not their own and either destroy the egg or abandon the nest. To translate this into an optimisation tool, Yang and Deb used three idealised rules:

1. Each turn cuckoo lays one egg (i. e. a potential solution) and dumps it in a randomly chosen nest (i. e. member of population)
2. A fraction of the nests containing the best eggs (i. e. the fittest members of population) will carry over to the next generation
3. The number of nests (i. e. population size) is constant and there is a probability (p_a) that a host can discover an alien egg, which can result in the abandoning of the nest

The last principle can be understood as follows. If the abandonment rate parameter p_a is set to 0.2, then 20% of worst nests will be replaced with the newly generated ones.

The steps involved in the CS are derived from the above-mentioned rules and are shown in Figure 4.1.

Cuckoo Search

Initialise a population P of m individuals $\mathbf{x}_i$, $P_m = (x_1, x_1, \ldots, x_m)$

For all $\mathbf{x}_i$ **do**
  Calculate fitness $F_i = f(\mathbf{x}_i)$
End for
While (ObjectiveEvaluationNumber < MaxEvaluationNumber)
  Create individual ($\mathbf{x}_j$) via Lévy Flight
  Calculate fitness $F_j = f(\mathbf{x}_j)$
  Choose random individual $\mathbf{x}_i$ from population P_m
  If ($F_j > F_i$) **then**
    Replace $\mathbf{x}_i$ with $\mathbf{x}_j$
  End if
  Abandon a fraction p_a of individuals with worst fitness
  Generate new random individuals
End while
Find the fittest individual

Figure 4.1 – CS pseudo-code

As can be noted from the above pseudo-code, an important aspect of the CS is the use of Lévy flight for both local and global searching. The Lévy flight process, which has previously been used in other search heuristics (Pavlyukevich, 2007), is a random walk that is characterised by a series of instantaneous moved chosen from a probability density function which has a power law tail. This process represents the optimum random search pattern and is frequently found in nature (Viswanathan, 2008).

When generating a new individual, a Lévy flight is performed starting at the position of the fittest individual of the population. If the objective function (i.e. fitness) of the new individual is better than the objective function of another randomly selected one, the new individual replaces it. The scale of this random search is controlled by multiplying the generated Lévy flight by a step size α:

$$x_i^{(t+1)} = x_i^t + \alpha \oplus Levy(\lambda) \tag{22}$$

For example, setting α = 0.1 could be beneficial for problems with a small domains, whereas setting it to a larger values makes the algorithm suitable for problems with bigger domains. Yang and Deb (2009) did not discuss the boundary handling in their formulation. Instead, they use an approach similar to PSO boundary handling (Yilmaz & Kuzuoglu, 2009): when a Lévy flight results in the generation of an individual outside the bounds of the objective function, the fitness and position of the original individual would not change.

One of the advantages of CS over other popular metaheuristics such as PSO and GA is that it needs only adjust one parameter – the abandonment rate p_a. Yang and Deb (2010) found that the convergence rate was not strongly affected by the value of p_a and they suggested setting it to p_a = 0.25. Moreover, as the result of experimental evaluations (Yang & Deb, 2010), the CS has been shown to perform well in comparison to GA and PSO.

4.1.2 Discrete Cuckoo Search for RCPSP

The areas of application of the original CS, as was intended by its creators, were problems in the continuous optimisation domain. Because of that, this version of the algorithm cannot be directly utilised to solve the RCPSP, as it is a combinatorial optimisation problem. In order to do so, the ideas of CS need to be extended to the discrete domain, thus DCS is presented.

One of the major goals of extending CS to solve the RCPSP is to retain its key advantages, such as little amount of parameters to configure and efficiency, and incorporate them into the discrete version of the algorithm. The adaptation of CS to the RCPSP primarily emphasises the reinterpretation of its key elements:

1. Solution representation scheme
2. Objective evaluation
3. Lévy flight

4.1.2.1 Solution Representation Scheme

The efficiency of a representation scheme is expected to be an important factor for performance of any algorithm. For the RCPSP it is more convenient to operate on an encoded solution (i. e. indirect form) rather than work with its direct form, as it is difficult to consider both precedence and resource constraints simultaneously when new individual is generated. In the scope of the RCPSP, the direct form of a solution is the resulted schedule of a project, whereas the indirect form of a solution is the encoded version of a schedule, designed to eliminate complexities of schedule manipulation for an algorithm.

According to Palmer and Kershenbaum (1994), in order for a metaheuristic algorithm to function properly, the ideal solution representation scheme must possess the following properties:

1. Computationally fast transformation of a representation into a solution
2. Each solution in the original space has a solution in the encoded space
3. Each encoded solution corresponds to one feasible solution in the original space
4. All solutions in the original space are represented by the same amount of encoded solutions
5. Small changes in the encoded solution result in the small changes in the original solution

Kolisch and Hartmann (1999) reviewed popular solution representation schemes and their appropriate operators for the RCPSP and outlined the two most commonly implemented and used ones: Activity List (AL) and Random Key (RK).

The AL representation scheme is a vector of size n, elements of are activities. Index of each of the AL's elements depicts the order in which an activity is going to be scheduled; hence, activities in the AL are scheduled in the same order as

they presented. The RK, on the other hand, encodes a solution as a vector of n numbers where the i^{th} number relates to the i^{th} activity. RK is transformed into a schedule by successively scheduling activities with highest random keys (random numbers). Based on computational experiments conducted on sets of benchmark instances from PSPLIB, Kolisch and Hartmann (2006) concluded that AL is more efficient than the RK and algorithms that operate on this representation scheme tend to produce better results. In the performance evaluation of more than 20 heuristics for the RCPSP, top 8 algorithms operated on AL representation scheme.

In order to demonstrate how AL representation scheme of a sample schedule would look like, an example project is taken from (Debels & Vanhoucke, 2005), as shown in Figure 4.2. The project consists of 19 non-dummy activities and single resource with capacity of 10 units. In the upper part of the figure, a project network is displayed. Under each node, a duration and resource request of the corresponding activity are provided respectively. In the lower part of the figure, a feasible schedule of makespan 47 is represented in the form of an extended Gantt chart. The horizontal axis of the chart shows time when each activity is executed, while the vertical axis shows a number of resources are taken. Each block on the chart corresponds to an activity from the sample project.

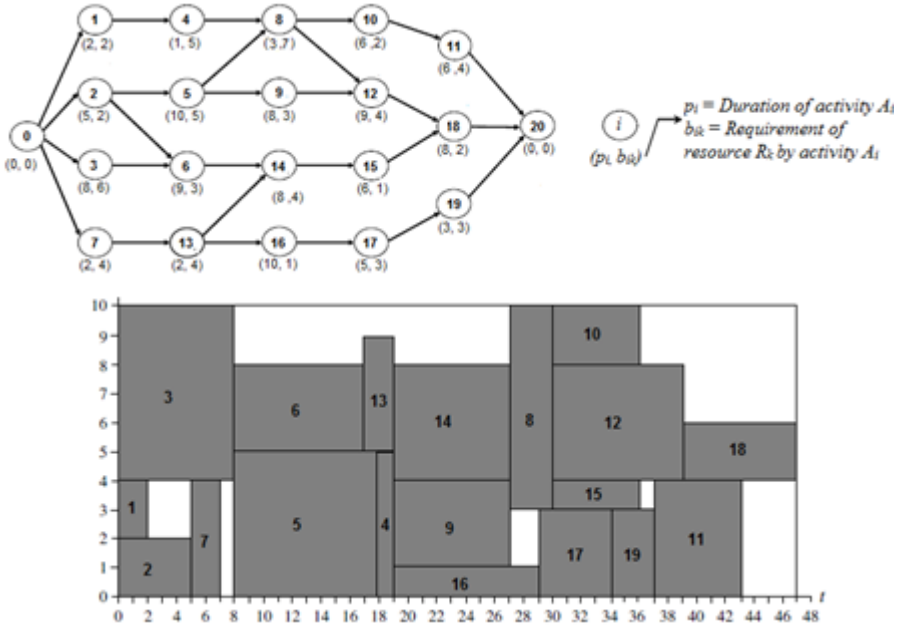

Figure 4.2 - Sample project network and schedule

The AL representation of the above schedule is presented in Figure 4.3. In the figure, an array consisting of 19 elements is shown. Each element in the array has a number which depicts the activity's number in the project. Further, below each element, its starting time is given.

1	2	3	7	5	6	13	4	9	14	16	8	17	10	12	15	19	11	18
0	0	0	5	8	8	17	18	19	19	19	27	29	30	30	30	34	37	39

Figure 4.3 - Activity List representation of a sample schedule

4.1.2.2 Objective Evaluation

In the RCPSP, in order to convert the solution representation scheme (i.e. AL) into schedule and estimate its makespan, the procedure called Schedule Generation Scheme (SGS) is applied. SGS constructs a schedule by scheduling each activity, one at a time, according to the sequence defined by the representation scheme. Kolisch (1996) divided SGS into two kinds: serial and parallel.

Serial SGS works as follows: at each iteration, an eligible activity is selected according to its priority (i.e. their position in the AL) and inserted inside a partial schedule at the earliest possible time (respecting the project precedence and resource constraints), while keeping unchanged the starting times of the already scheduled activities. An activity is eligible if all its predecessors have been scheduled. Parallel SGS, on the other hand, schedules a set of activities at each iteration. With each iteration i it associates schedule time t_i, which equals to the latest finishing time of the already scheduled activities at time t_{i-1}. The activities, which are available for scheduling with respect to precedence and resource constraints, are scheduled at t_i one by one with respect to their priority order. Once this is done, the next schedule time and related set of eligible activities are computed. This is repeated until activities are scheduled.

As was demonstrated by Sprecher et al. (1995), schedules that are generated by serial SGS are referred to as active, whereas schedules that are created by parallel are referred to as non-delay. Kolisch (1996) has shown that a set of active schedules will always contain an optimal schedule (hence the name). On the other hand, the schedules produced by parallel SGS are able to utilise resources as early as possible (without delays), leading to schedules that are more compact. Further, in his study of both variants of SGS, Kolisch concluded that on average, in terms of obtaining optimal results and overall success rate, serial

SGS performs better than parallel SGS, espcially on instances which consists of many activities and/or scarce resource capacties.

Based on the above-mentioned research and conclusion from the author (Kolisch R. , 1996), serial SGS is selected as the main measure of evaluation of the objective. The pseudo-code of serial SGS is presented in Figure 4.4.

```
Serial Schedule Generation Scheme
Initialise activity list A of size n, An = (a1, a2, …, an)
Initialise empty set of activity starting times S

For all ai do
   Find earliest possible starting time si by checking finishing time
      of its predecessors
   While(resourceConstraintsNotSatisfied(ai, si))
      si++
   End while
   S[ai] = si
End for
```

Figure 4.4 - Serial Schedule Generation Scheme pseudo-code

Serial SGS converts given activity list A by scheduling each of its activities as early as possible in the same order as they appear. Concretely, the whole process can be broken down into the following steps:

- Pick first unscheduled activity a_i from A
- If a_i has any predecessors, find the latest finishing time s_i
- Check resource availabilities at time s_i. If starting activity ai is not possible, increment s_i until resource constraints are not satisfied
- Once a_i is scheduled, proceed with the next activity in A

4.1.2.3 Lévy Flight

Yang and Deb (2009) have demonstrated that Lévy flights improve search for solutions in continuous optimisation problems and enhance the overall performance of the algorithms. Moreover, they were able to show that Lévy flights are characterised by intensive search around local solution followed by occasional big steps in the long run.

Several ways of Lévy flight implementations exist. Leccardi (2005) compared different approaches for generation of Lévy flight values and as the conclusion of his experiments, he estimated that the algorithm developed by Mantegna (1994)

is the most efficient method. Mantegna's (1994) algorithm produces random noise according to a symmetric Lévy distribution, which is ideal for Lévy flight.

In the Mantegna's (1994) algorithm, the Lévy distribution is calculated as

$$Levy(\lambda) = \frac{u}{|v|^{\frac{1}{\lambda}}} \tag{23}$$

where u and v are drawn from normal distributions defined as

$$u \sim N(0, \sigma_u^2) \qquad v \sim N(0, \sigma_v^2) \tag{24}$$

$$\sigma_u = \frac{\Gamma(1+\lambda) * \sin(\frac{\pi\lambda}{2})}{\Gamma(\frac{1+\lambda}{2}) * \lambda * 2^{\frac{\lambda-1}{2}}} \qquad \sigma_v = 1 \tag{25}$$

where the distribution parameter $\lambda \in [0.3, 1.99]$ and Γ denotes Gamma function.

To adapt Lévy flight to a problem in the discrete domain, the Lévy distribution number generated by Mantegna's algorithm is associated with the amount and types of operations (i.e. steps) that will be performed on an individual from the population in an attempt to transform it into a better one.

Depending on the range to which the received value belongs, the following operations can be performed:

1. [0, i] – perform one small step
2. [(k-1) *i, k * i] – perform k amount of small steps
3. [k * i, 1] - perform big step

where the value of i in this process is i = 1/(s+1), s is a configurable parameter representing the max number of steps that can be performed and $k \in [2, .., s]$.

For example, assume that s = *4*, hence i = 0.2, therefore the whole interval is divided into five parts:

1. Lévy in [0, 0.25] – small step
2. Lévy in [0.25. 0.5] – 2 small steps
3. Lévy in [0.5, 0.75] – 3 small steps
4. Lévy in [0.75, 1] – big step

The step of a movement represents a distance in the search space that will be travelled to obtain a new solution by application of mutation operators.

To mimic a small step, the simple shift operator (Della Croce, 1995) is applied. The simple shift operator randomly selects an activity a_i inserts it immediately

after another activity a_i, given that precedence constraints are not violated. In the example in Figure 4.5, sample AL from Figure 4.3 is used to demonstrate how this operator works: here Activity 14 is selected and inserted right after Activity 17.

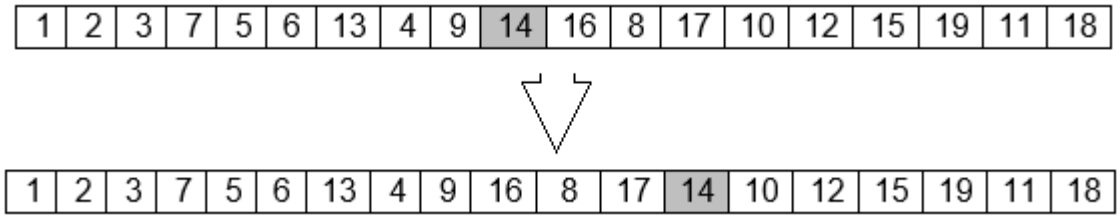

Figure 4.5 - Shift operator example

To mimic a large step, pairwise interchange operator (Hartmann, 1998) is applied. Pairwise interchange is defined as swapping two randomly-picked activities a_i and a_j if in the resulting activity list precedence constraints are not violated. In the example shown in Figure 4.6, Activity 16 is swapped with Activity 12.

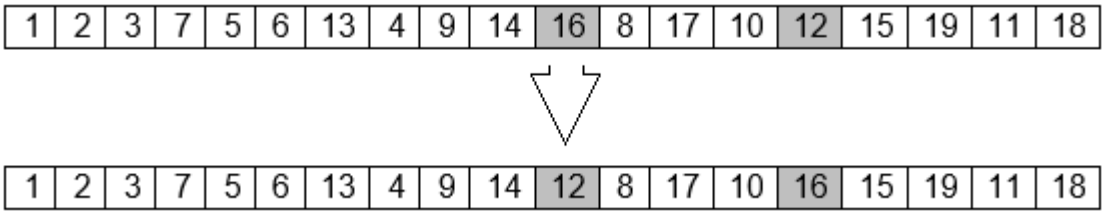

Figure 4.6 - Pairwise Interchange operator example

4.1.3 Computational Performance

To evaluate the performance of the DCS, various numerical experiments are conducted on sets of benchmark instances from PSPLIB (Sprecher, Kolisch, & Drexl, 1995) designed specifically for testing RCPSP methodologies. PSPLIB contains instances of scheduling problems with varying difficulty, which are grouped into sets in accordance to the amount of activities each project contains. The following sets are available:

- J30 – 480 instances of scheduling problems, each consisting of 30 activities and 4 resource types
- J60 – 480 instances of scheduling problems, each consisting of 60 activities and 4 resource types

- J120 – 600 instances of scheduling problems, each consisting of 120 activities and 4 resource types

Due to the complexity of the problem, the optimal solutions are only available for J30 set, whereas for J60 and J120 instances only best-known solutions are given.

In order to assess performance of an algorithm after running each of the benchmark instances, a deviation from optima is calculated:

$$deviation = \frac{ms_r - ms_o}{ms_o} * 100\% \quad (26)$$

where ms_r is the makespan of received solution, whereas ms_o is the makespan of optimal solution.

For J30 instances, deviation is calculated with respect to optimal solutions, while for J60 and J120 instances deviation is calculated with respect to the length of the critical path (CP). CP is obtained by computing the makespan of a project by relaxing the resource constraints of the problem (Hartmann & Briskorn, 2010).

4.1.3.1 Experimental Setup

Before the performance of the algorithm can be evaluated and compared with others, it is necessary to configure it and find the most appropriate parameters setting. To do this, the *irace* package (Birattari, Yuan, Balaprakash, & Stützle, 2010) is utilised in the experimental setup.

The *irace* package is an automatic configuration tool for tuning optimisation algorithms, that is, automatically finding good configurations for the parameters values. *Irace* works by receiving a list of algorithm's parameters as input and uses a set of training instances to find the optimal levels for each of the parameters. This is achieved by searching in the parameter search space for good performing algorithm configurations by executing the target algorithm on different instances with different parameter configurations.

In this experimental setup, *irace* is set to use benchmark instances from PSPLIB to tune the algorithm. In this setup benchmark instance from J30, J60, and J120 sets have been utilised for tuning of the target algorithm:

- J30 set – every tenth instance starting from number 1, 48 instances total
- J60 set – every tenth instance starting from number 1, 48 instances total

- J120 set – every tenth instance starting from number 1, 60 instances total

The stopping criterion for running each of the instances was set to 5000 objective evaluations.

After the algorithm is configured and optimal parameters are identified, its performance can be evaluated and compared against other methodologies. It is assumed that these other methodologies would have already gone through the same parameter fine-tuning process. Typically, performances of algorithms for the RCPSP are evaluated by running all benchmark instances from J30, J60, and J120 sets from PSPLIB. In order to provide the basis for comparison with other algorithms, Hartmann et al. (2000) suggested to limit the execution of algorithms to the amount of times the objective function is evaluated (i. e. the number of generated schedules). The advantage of this stopping criterion is that it is independent of the computer platform. Therefore, all heuristics can be tested using the original implementation and the best configuration. Moreover, such tests are independent of compilers and implementation skills, thus the concept of algorithm is evaluated, rather than its program code. Hence, in order to evaluate the performance of an algorithm, three sets of experiments have been conducted in which the algorithm will have to run all benchmark instances from J30, J60, and J120 sets for the three stopping criteria (maximum of 1000, 5000, and 50000 objective function evaluations).

4.1.3.2 Parameters Settings

The DCS has three configurable parameters:

- Population size m
- Abandonment rate p_a
- Max amount of steps s

In order to find the optimal values for these parameters, a sensitivity analysis has been carried out using the *irace* package. Ranges of parameters value selected for the analysis are summarised in Table 4.1.

Table 4.1 - DCS parameter values for sensitivity analysis

Parameter	Values Range
Population size (m)	[10, 200]
Abandonment rate (p_a)	[0, 0.9]
Max amount of steps (s)	[1, 10]

As the result of the algorithm tuning, the optimal parameters values identified by *irace* are summarised in Table 4.2.

Table 4.2 - DCS optimal parameters values

Parameter	Value
Population size (m)	18
Abandonment rate (p_a)	0.8
Max amount of steps (s)	4

During the tuning process, *irace* iteratively updated the sampling models of the parameters to focus on the best regions of the parameter search space. The frequency of the sampling of parameters values in the regions of the specified parameters search space for m, p_a and s is presented in graphs in Figure 4.7, Figure 4.8 and Figure 4.9.

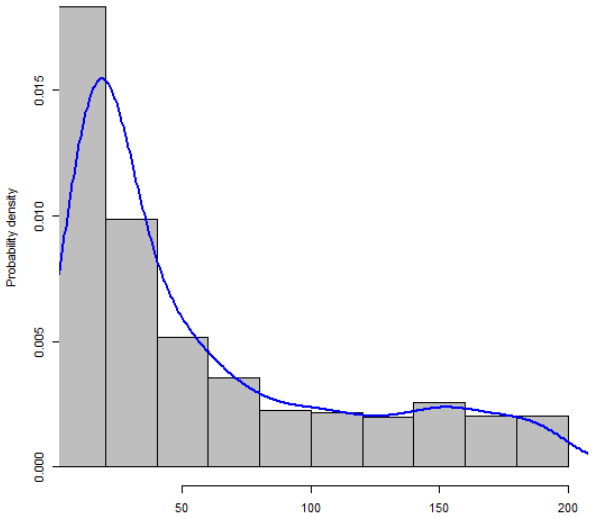

Figure 4.7 - Population size sampling frequency

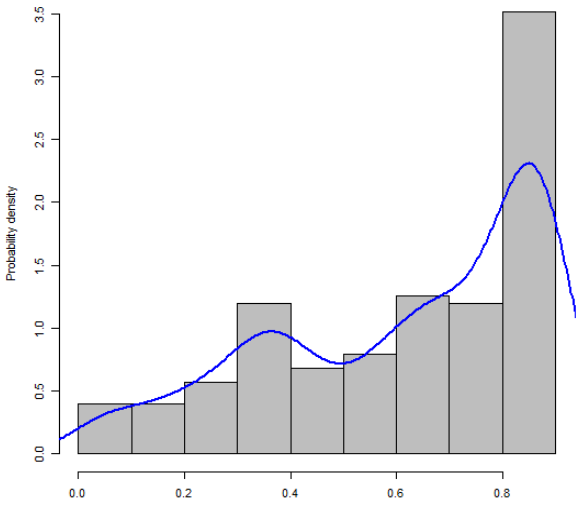

Figure 4.8 - Abandonment rate sampling frequency

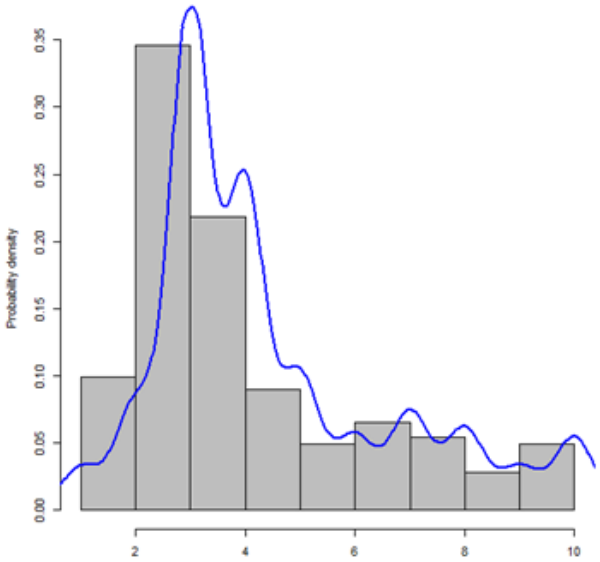

Figure 4.9 - Max amount of steps sampling frequency

The graph in Figure 4.10 displays the interaction between parameters and their dependencies on one another on the example of 100 best parameters configurations obtained by *irace* package during fine-tuning.

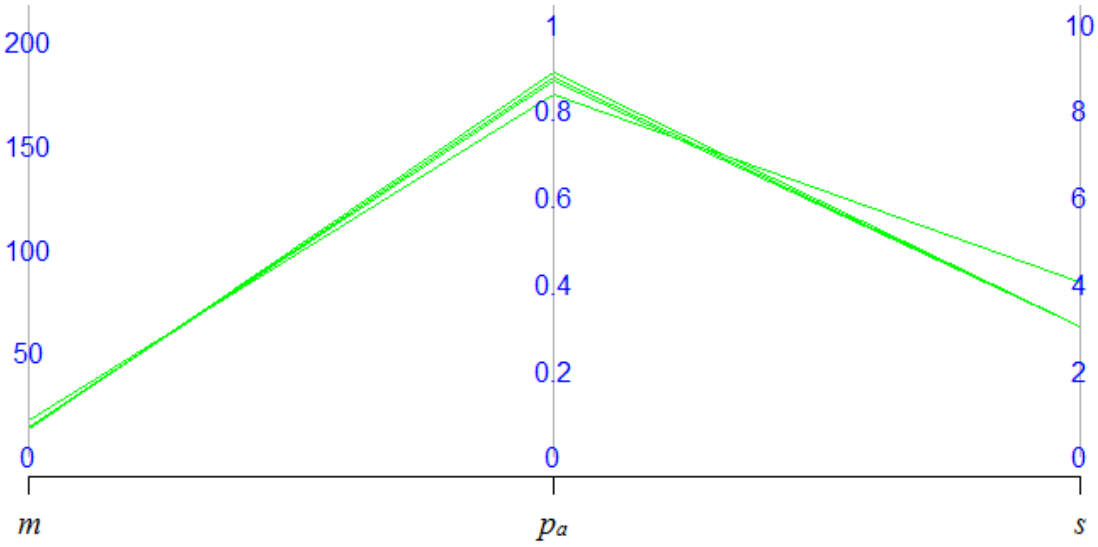

Figure 4.10 - DCS parameters correlations

During the fine-tuning process, *irace* iteratively updated the sampling models of the algorithm's parameters to focus on the best regions of the parameter search space. The frequency of the sampled configurations presented in Figure 4.7, Figure 4.8 and Figure 4.9 provide insights on the parameter search space of DCS. From these graphs it is possible to understand what the optimal parameters are and how changing their value might affect the overall performance of the algorithm.

Graph in Figure 4.10 is a parallel coordinates plot in which each of its axes corresponds to the algorithm's parameter, whereas green lines on the plot correspond to the parameter configurations that were obtained by *irace* package. The presented graph helps to visualise what kind of parameter configurations where used the most during the algorithm fine-tuning. In this particular example, the values of 100 best-obtained parameter configurations are plotted. From the presented in it is easy to see that parameter values of all best configurations are in regions of [10; 20] for m, [0.8; 0.9] for p_a and [3; 4] for s, meaning that there is almost no ambiguity between the best-obtained configurations

As can be observed from the above-presented graphs, the optimal levels of DCS parameters were in the following ranges:

- m – [10; 20]
- p_a – [0.8; 0.9]
- s – [3; 4]

Due to high abandonment rate, the overall population is constantly updated with new individuals, hence, there is no need for having maintaining high population. By keeping max amount of step parameter s values between 2 and 4, the algorithm provides perfect balance between usage of pairwise interchange and shift operators.

4.1.3.3 Comparative Analysis

The computational results of the performance evaluation of DCS are presented in Table 4.3, Table 4.4, and

Table 4.5 for J30, J60 and J120 instance sets, respectively. The first column "Algorithm" reports abbreviations of the algorithms considered for comparison. Column "Author(s)" reports the name(s) of the original author(s) and reference to the work in which the algorithm at hand was previewed. The last column refers to the average deviation % for three stopping conditions: 1000, 5000 and 50000 objective evaluations. For the J30 instances, the average deviation is shown with respect to optimal solutions, while for J60 and J120 instances the average deviation % is calculated with respect to the CP length. Computational performance of other presented algorithms was taken from (Hartmann & Kolisch, 2000) and (Kolisch & Hartmann, 2006).

Table 4.3 - DCS performance comparison for J30 set

Algorithm	Author(s)	Dev (%)		
		1000	5000	50000
DCS	Bibiks et a.	0.52	0.16	0.05
TS	Nonobe and Ibaraki (2002)	0.46	0.16	0.05
GA	Hartmann (1998)	0.38	0.22	0.08
Sampling + BF	Tormos and Lova (2001)	0.30	0.17	0.09
ANGEL	Tseng and Chen (2006)	0.22	0.09	n/a

Table 4.4 - DCS performance comparison for J60 set

Algorithm	Author(s)	Dev (%)		
		1000	5000	50000
DCS	Bibiks et al.	12.89	12.46	11.18
GA	Hartmann (1998)	12.21	11.70	11.21
Sampling + BF	Tormos and Lova (2001)	11.88	11.62	11.36
ANGEL	Tseng and Chen (2006)	11.94	11.27	n/a
TS	Nonobe and Ibaraki (2002)	12.97	12.18	11.58

Table 4.5 - DCS performance comparison for J120 set

Algorithm	Author(s)	Dev (%)		
		1000	5000	50000
DCS	Bibiks et al.	37.91	35.29	33.20
GA	Hartmann (1998)	37.19	35.39	33.21
Sampling + BF	Tormos and Lova (2001)	36.24	35.56	34.77
ANGEL	Tseng and Chen (2006)	36.39	34.49	n/a
TS	Nonobe and Ibaraki (2002)	40.86	37.88	35.85

The performance evaluation of DCS algorithm can be regarded as satisfactory. By being capable of outperforming GA and TS, among others, DCS shows acceptable level of performance which can be compared with other non-hybrid metaheuristics.

4.2 Discrete Flower Pollination Algorithm

Flower pollination algorithm (FPA) is a metaheuristic algorithm developed by Yang (2012) and it represents a generalised version of the CS. Similarly to CS, FPA is a population-based metaheuristic that uses Lévy flight to explore solution

search space. The main different between the two is the inclusion of the crossover operator in the latter one. As with CS, FPA originally was developed for application in continuous optimisation problems. Thus, the modification, namely discrete flower pollination algorithm (DFPA), for application in the RCPSP is proposed. The main reason for the implementation of this algorithm in the context of this PhD project is to analyse the impact of addition crossover operator to the original paradigm of CS.

4.2.1 Flower Pollination Algorithm

As witch the CS, the ideas of FPA were inspired by a nature process, which in this case is the reproduction of flowers. Based on the characteristics of flower pollination process in nature, Yang (2012) based the algorithm on the following rules:

1. Two types of pollination processes are considered: local pollination and global pollination
2. Switching between local and global pollination is controlled by probability $p_s \in [0, 1]$
3. During local pollination, a new solution is generated via Lévy flight
4. During local pollination, a new solution is generated via application of crossover operator

Based on the above definition, the algorithm's pseudo-code is summarised in Figure 4.11.

Flower Pollination Algorithm

Initialise a population P of m host flowers $\mathbf{x}_i$, $P_m = (x_1, x_{1,} \ldots, x_m)$

For all $\mathbf{x}_i$ **do**
  Calculate fitness $F_i = f(\mathbf{x}_i)$
End for
While (ObjectiveEvaluationNumber < MaxEvaluationNumber)
  Generate random number *r* in range [0, 1]
  If ($r < p_s$) **then**
    Create individual ($\mathbf{x}_j$) via Lévy flight
  Else
    Create individual ($\mathbf{x}_j$) via crossover operator
  End if
  Calculate fitness $F_j = f(\mathbf{x}_j)$
  Choose random individual $\mathbf{x}_i$ from population P_m
  If ($F_j > F_i$) **then**
    Replace $\mathbf{x}_i$ with $\mathbf{x}_j$
  End if
End while
Rank all individuals and find the fittest nest

Figure 4.11 – FPA pseudo-code

As can be noted from the pseudo-code in Figure 4.11, FPA offers two ways of creation of new individuals. If the value of a randomly drawn number *r* in the range of [0, 1] is less than the value of switching probability p_s, then, as in CS, a new individual x_j is generated via application of Lévy flight in accordance to (21). Otherwise, a crossover operator is utilised.

4.2.2 Discrete Flower Pollination Algorithm for RCPSPs

Previously, FPA have not been applied to problems in the discrete domain. Similarly to DCS, application of FPA to such problems focuses on the reinterpretation of its key elements and operators:

- Solution representation scheme
- Objective evaluation
- Lévy flight
- Crossover operator

In DFPA, the reinterpretation of solution representation scheme, objective evaluation and Lévy flight follow the same procedure as described in sections 4.1.2.1, 4.1.2.2 and 4.1.2.3, respectively.

To implement global pollination in a discrete domain, two-point crossover operator is used (Hartmann, 1998), also known as Davis order crossover. Given that there are two parents (parent 1 and parent 2), two-point crossover works by drawing two random integers q_1 and q_2 with $1 < q_1 < q_2 < n$, where n is the total amount of activities in the AL. Further the offspring is created by taking the activity sequence of position $i = q_1, \ldots, q_2$ from parent 1 and the remaining activities from parent 2 in the same order as they appear. An example of this procedure is demonstrated in Figure 4.12 on the sample AL from Figure 4.3.

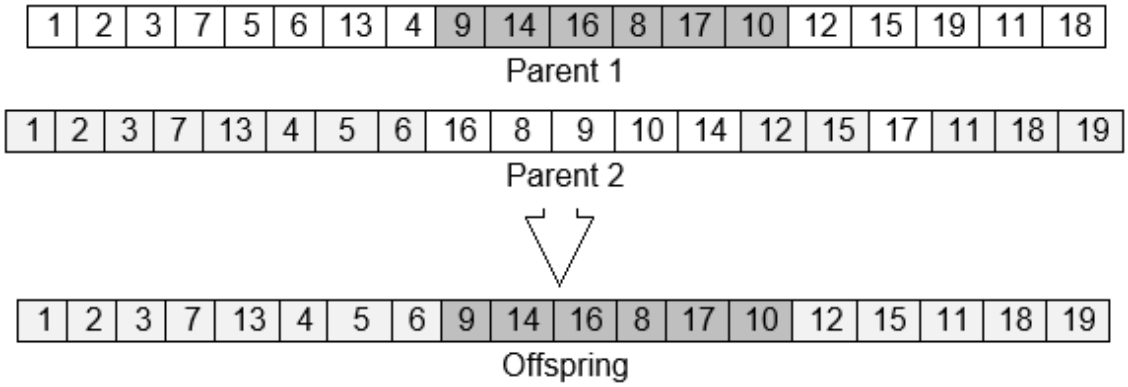

Figure 4.12 - Two-point Crossover example

4.2.3 Computational Performance

Similarly to DCS, performance of DFPA has been evaluated by running each of the benchmark instances from PSPLIB. The results of evaluation are then compared with other heuristic for RSPCP from (Hartmann & Kolisch, 2000) and (Kolisch & Hartmann, 2006). Description of the full experimental setup for parameters tuning and performance evaluation can be found in Section 4.1.3.1.

4.2.3.1 Parameters Settings

The DFPA has three configurable parameters:

- Population size m
- Switching probability p_s
- Max amount of steps s

In order to find the optimal values for these parameters, a sensitivity analysis has been carried out using the *irace* package. The ranges of parameters values selected for the analysis are summarised in .

Table 4.6.

Table 4.6 – DFPA parameter values for sensitivity analysis

Parameter	Values Range
Population size (m)	[10, 200]
Switching probability (p_s)	[0, 0.9]
Max amount of steps (s)	[1, 10]

As the result of the algorithm tuning, the optimal parameters values identified by *irace* are summarised in Table 4.7.

Table 4.7 - DFPA optimal parameters values

Parameter	Value
Population size (m)	116
Switching probability (p_s)	0.8
Max amount of steps (s)	8

During the tuning process, *irace* iteratively updated the sampling models of the parameters to focus on the best regions of the parameter search space. The frequency of the sampling of parameters values in the regions of the specified parameters search space for m, p_s and s is presented Figure 4.13, Figure 4.14 and Figure 4.15.

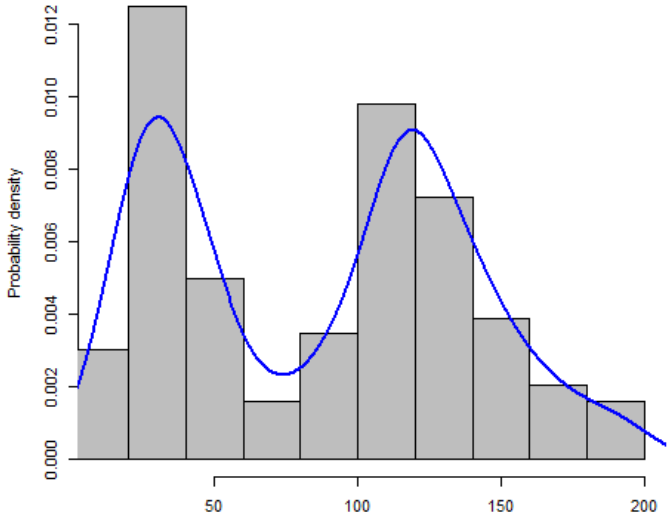

Figure 4.13 - Population size sampling frequency

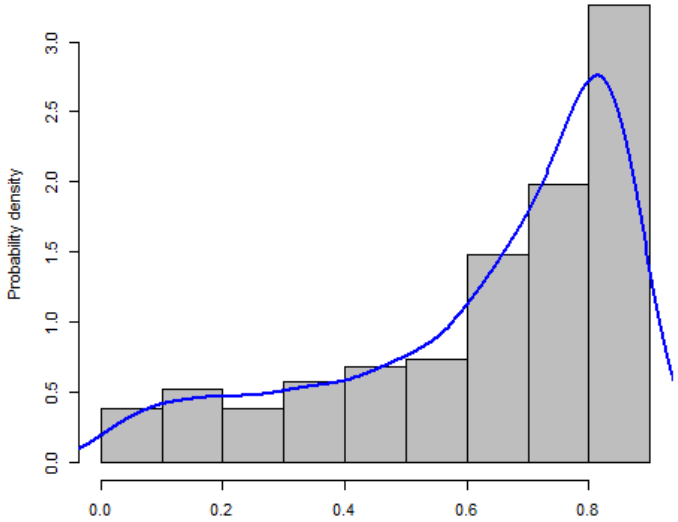

Figure 4.14 - Switching probability sampling frequency

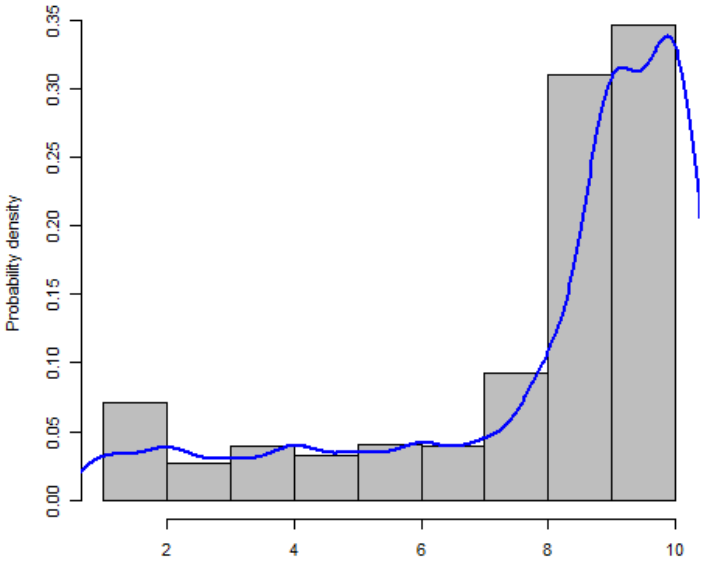

Figure 4.15 - Max amount of steps sampling frequency

The graph in Figure 4.16 displays the interaction between parameters and their dependencies on one another on the example of 100 best parameters configurations obtained by *irace* package during fine-tuning.

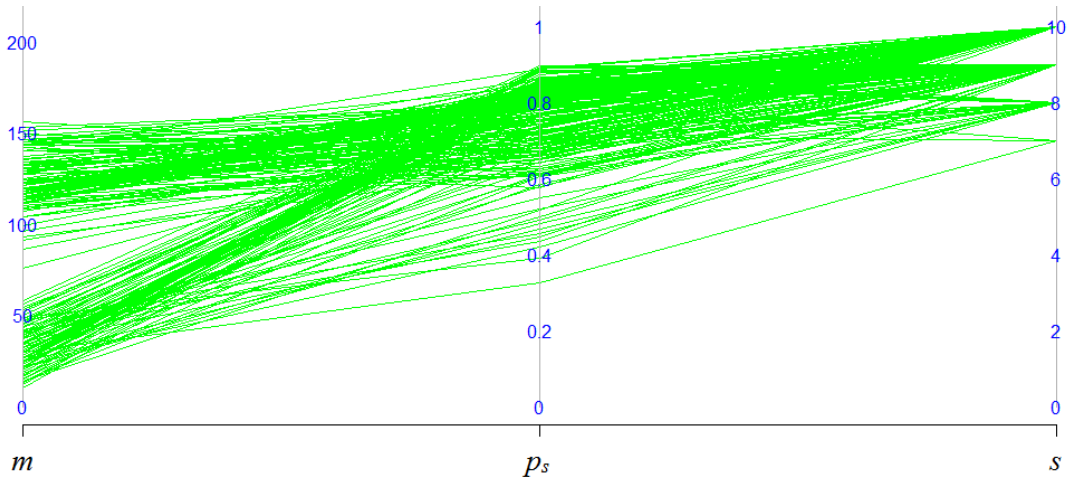

Figure 4.16 - DFPA parameters correlations

Unlike with DCS, by looking at the graph in Figure 4.16 it is easy to see that DFPA permits more combinations of optimal parameters settings. The optimal levels of DFPA parameters identified by *irace* package were in the following regions:

- m – [10; 50] & [100; 150]
- p_s – [0.4; 0.9]
- s – [7; 10]

4.2.3.2 Comparative Analysis

The computational results of the performance evaluation of DFPA are presented in

Table 4.8, Table 4.9 and Table 4.10 for J30, J60 and J120 instance sets, respectively. The first column “Algorithm” reports abbreviations of the algorithms considered for comparison. Column “Author(s)” reports the name(s) of the original author(s) and reference to the work in which the algorithm at hand was previewed. The last column refers to the average deviation % for three stopping conditions: 1000, 5000 and 50000 objective evaluations. For the J30 instances, the average deviation is shown with respect to optimal solutions, while for J60 and J120 instances the average deviation % is calculated with respect to the CP length. Computational performance of other presented algorithms was taken from (Hartmann & Kolisch, 2000) and (Kolisch & Hartmann, 2006).

Table 4.8 - DFPA performance comparison for J30 set

Algorithm	Author(s)	Dev (%)		
		1000	5000	50000
TS	Nonobe and Ibaraki (2002)	0.46	0.16	0.05
DFPA	Bibiks et a.	0.46	0.19	0.06
GA	Hartmann (1998)	0.38	0.22	0.08
Sampling + BF	Tormos and Lova (2001)	0.30	0.17	0.09
ANGEL	Tseng and Chen (2006)	0.22	0.09	n/a

Table 4.9 - DFPA performance comparison for J60 set

Algorithm	Author(s)	Dev (%)		
		1000	5000	50000
DFPA	Bibiks et al.	13.01	12.90	11.20
GA	Hartmann (1998)	12.21	11.70	11.21
Sampling + BF	Tormos and Lova (2001)	11.88	11.62	11.36
ANGEL	Tseng and Chen (2006)	11.94	11.27	n/a
TS	Nonobe and Ibaraki (2002)	12.97	12.18	11.58

Table 4.10 - DFPA performance comparison for J120 set

Algorithm	Author(s)	Dev (%)		
		1000	5000	50000
GA	Hartmann (1998)	37.19	35.39	33.21
DFPA	Bibiks et al.	37.82	35.55	33.46
Sampling + BF	Tormos and Lova (2001)	36.24	35.56	34.77
ANGEL	Tseng and Chen (2006)	36.39	34.49	n/a
TS	Nonobe and Ibaraki (2002)	40.86	37.88	35.85

As can be noted from the above-presented results, the addition of the crossover operator to the algorithm's paradigm did not have much of an impact on its overall performance. In all tests, DFPA has showed weaker results than DCS.

4.3 Improved Discrete Cuckoo Search

Since DCS and DFPA structurally are very similar algorithms, they share the same common pitfalls and limitations, which are specific to the chosen solution

representation scheme, namely AL, and utilised operators. These limitations can be summarised as follows:

- Representation of one schedule by several structurally different ALs
- Random and context-unaware operators

The first limitation described above is only specific to the AL. Due to the characteristics of this solution representation scheme, for one schedule several different representations can exist. Because of that, in some cases, modification of one AL can result in the generation of exactly the same schedule, hence, the wasted objective evaluation. Eliminating this limitation will allow an algorithm to perform better and with less iterations.

The second limitation is specific to the operators used in both algorithms: shift operator, pairwise exchange, and two-point crossover. Even though these operators are easy to implement, their operation is purely based on a random factor and it does not take into consideration any specifics of the problem setting. Improving these algorithms by adding some elements of intelligence to them will most certainly increase the algorithm's efficiency and success rate.

As an attempt of addressing the above-mentioned inefficiencies of DCS and DFPA and improving their overall performance the improved discrete cuckoo search (IDCS) is proposed.

4.3.1 Improved Discrete Cuckoo Search for RCPSPs

IDCS represents a modified version of the earlier-presented DCS and its main objective is to resolve the weaknesses that were described earlier. The main additions to the original paradigm of the DCS are shown in bold in Figure 4.17 and can be summarised as follows:

- New solution representation scheme
- Improved Lévy flight
- Improvement of fraction of individuals via local search
- Use of crossover operator

Improved Discrete Cuckoo Search

Initialise a population P of m host nests $\mathbf{x}_i$, $P_m = (x_1, x_1, \ldots, x_m)$

For all $\mathbf{x}_i$ **do**
  Calculate fitness $F_i = f(\mathbf{x}_i)$
End for
While (ObjectiveEvaluationNumber < MaxEvaluationNumber)
  Create individual ($\mathbf{x}_j$) via Lévy flight
  Calculate fitness $F_j = f(\mathbf{x}_j)$
  Choose random individual $\mathbf{x}_i$ from population P_m
  If ($F_j > F_i$) **then**
    Replace $\mathbf{x}_i$ with $\mathbf{x}_j$
  End if
  Improve fraction p_c of individuals via local search
  Abandon a fraction p_a of individuals with worst fitness
  Generate new random individuals
End while
Find the fittest individual

Figure 4.17 - IDCS pseudo-code

4.3.1.1 Solution Representation Scheme

Both AL and RK schemes can have several representations for a single schedule. Because of that property, the solution space contains worthless representations that can be converted into identical schedules, thus reducing efficiency of an algorithm and resulting in wasted objective evaluations. The essential issues of the discussed representation schemes can be summed up as follows:

1. If two different RK schemes represent a linear combination of each other, then both schemes may result in the same solution
2. RK does not take into consideration precedence constraints
3. If two activities have identical starting times, then interchanging their position (for AL) and priority values (for RK) will not bring any changes to a solution
4. Because of the precedence constraints, activity positions (for AL) or priority values (for RK) will not have an effect on the starting time

Several attempts in the literature have been made to resolve the aforementioned inefficiencies. Debels et al. (2006) proposed a specialised RK representation scheme to tackle the above deficiencies by applying a transformation mechanism. To address the first issue, the authors utilised a

scaling of the Euclidian space, whereas for the second issue they created a repairing mechanism that considered the precedence constraints. Moreover, to address the third issue, activities with the same starting times were assigned the same priority values. To overcome the last issue, the activities were scheduled in topological order.

Moumene and Ferland (2008) proposed to decode solutions in a form of activity set list (ASL). ASL represents an ordered list of different non-empty subsets of activities. Each subset is comprised of activities that share common project characteristics, such as predecessors and successors. Because of the properties of ASL, the search space is significantly reduced, its exploration is more efficient and optimal solutions are never excluded.

Paraskevopoulos et al. (2012) proposed an alternative to ASL, called event list (EL). Similarly to ASL, the authors proposed to group activities in the EL with the same starting times into sets referred to as events. The EL representation scheme does not use any adjustments and repairing mechanisms and at the same time manages to effectively resolve the aforementioned inefficiencies of other representation schemes. Two different ELs can be differentiated by the distinct events that they contain, both in terms of structure and starting times. Thus, that way two different ELs cannot result in the same schedule. Moreover, the EL has been developed to allow local moves of sets of activities, enabling local search methodologies to generate newly enriched solution neighbours.

Figure 4.18 demonstrates solution representations of a sample schedule from Figure 4.2 using both AL and EL encodings in parts a) and b) of the image, respectively.

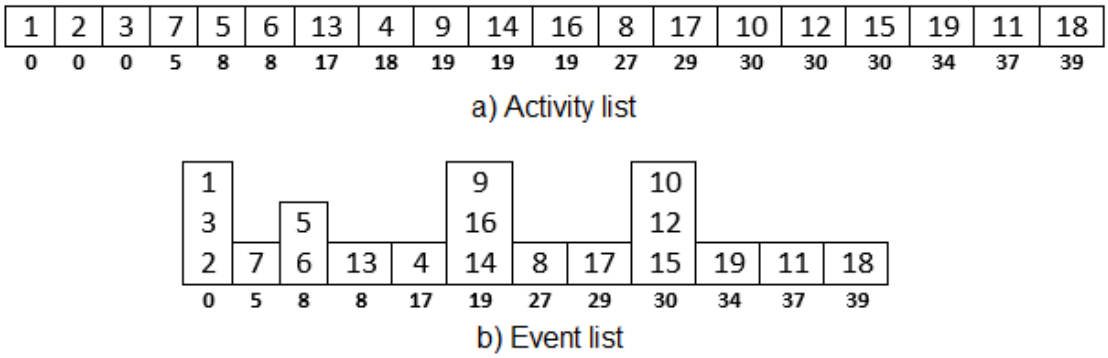

Figure 4.18 - Comparison of Activity List and Event List representations

As can be noted from the presented example, in EL activities with the same starting times are grouped together and the resulting groups are ordered by their

starting times. Each group of activities (with the same starting times) is considered as an event. Therefore, a schedule (or a solution) is a set of events ordered by their starting times.

4.3.1.2 Improved Lévy Flights

As in the original implementation of Lévy flight, the step of a movement represents a distance in the search space that will be travelled to obtain a new solution. However, this time small steps are proportional to a number of events that will be relocated by event move operator, whereas big step is mimicked by applying event crossover operator. When an event move is performed on an individuals, a new solution is obtained by relocation of its randomly picked events. The number of relocated events is proportional to the amount of small steps that needs to be performed (e.g. 1 step corresponds to the relocation of 1 event). When event crossover is performed, a new solution is obtained by merging two individuals from the population together. Since the application of event crossover symbolises a big step, the current individual is crossed with the one that is farthest from it.

To facilitate a control of movement through the search space, the amount of steps to be performed is associated with the Lévy distribution value generated by Mantegna algorithm, details of which are presented in Section 4.1.2.3.

Event move. The idea of event move operator, first proposed by Paraskevopoulos et al. (2012), is based on the utilisation of properties of the EL representation scheme. An event in EL represents a set of activities with equal starting times. Moreover, these activities can also share the same project characteristics (i. e. have matching predecessors and/or successors) and thus can be considered as one entity. However, this conjecture is not always true. In some situations, activities might be started at the same time because of the restrictions on the resource loads. Nevertheless, cases where activities have common network characteristics can be taken as an advantage, as this would allow to move a whole event throughout the schedule to a different position and, thus, generating a new, possibly better, solution. In that case, a larger amount of moves is permitted, which enriches the solution search space with more valuable solutions. If activities that form an event do not have any common network characteristics, then permitted local moves will be limited. The algorithm of event move is summarised in pseudo-code in Figure 4.19.

Event Move

Initialise event list E of size k, $E_k = (e_1, e_2, \ldots, e_k)$

Randomly pick an event $e_i = (a_1, a_2, \ldots, a_n)$ from E_k
For all $\mathbf{a}_j$ in e_i **do**
  Find allowable range of positions for relocation for a_j
  Relocate a_j to new position
End for
Apply SGS for objective evaluation

Figure 4.19 - Event Move pseudo-code

When event move is performed on an EL, a number of random events is picked for relocation. The number of picked events is proportional to the amount of steps that will be performed and one step corresponds to the relocation of one event. For each activity in the chosen event(s), a range of possible positions for relocation is calculated. The range of positions is formulated in accordance with the precedence relations between activities: positions of the latest starting predecessor and the earliest starting successor. Further, all activities that comprise the chosen event are moved independently from each other to a random positions within their allowable ranges. Depending on the situation, these activities might be added to already existing events or form a new event, if no suitable event exists. Note that events that consist only of one activity are considered for relocation as well.

After relocating events to new positions is complete, serial SGS is utilised to convert it into a schedule and calculate its makespan. It is worth mentioning that relocating event to a particular position in the EL does not guarantee that it will inherit its starting time. Instead, all of its activities will be rescheduled independently as early as possible and, as the result, starting times might change.

Figure 4.20 and Figure 4.21 demonstrate a schedule that has resulted from the event move. The project from Figure 4.2 is used as the initial solution.

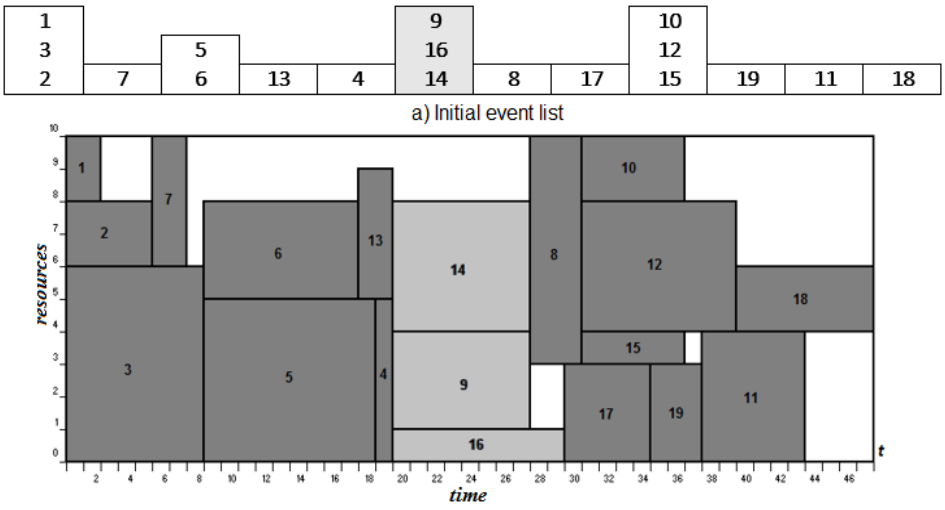

Figure 4.20 - Event Move example. Part 1

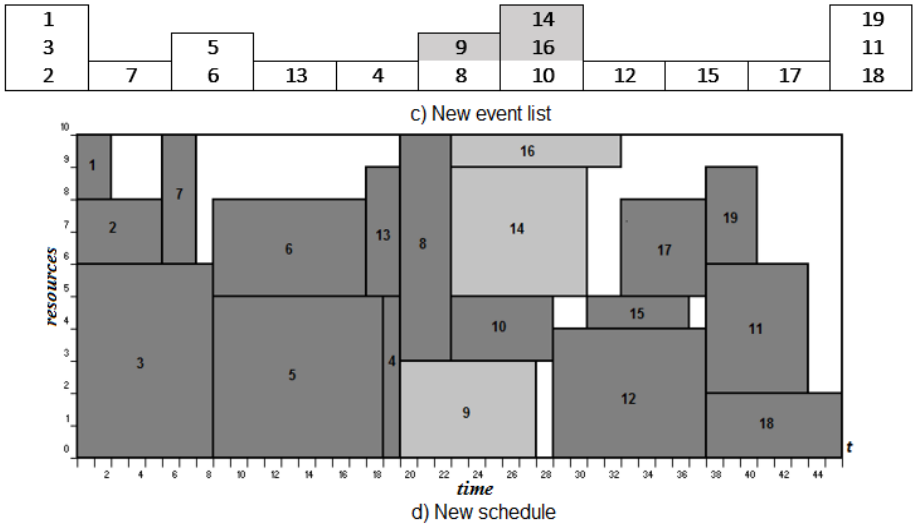

Figure 4.21 - Event Move example. Part 2

The process of event move begins with a selection of random event and estimation of feasible positions for relocation for each of the activities in the selected event. In the example shown in Figure 4.20a, the candidate event for relocation is highlighted and consists of Activities 9, 14 and 16. Further, the event is removed from the EL and all events that were scheduled before the chosen event remained at the same time slots. Subsequently, the activities from the removed event are inserted into the sub-solution at random positions within the allowable ranges. In the presented example, Activity 9 is inserted between Activities 8 and 10, while Activities 14 and 16 were inserted between Activities 10 and 12. As mentioned earlier in this section, even though local move relocates the events, during conversion of the EL into a schedule each activity is scheduled independently. Finally, the remaining activities are scheduled and the resulting

EL is given in Figure 4.21a, while its schedule is presented in Figure 4.21b. As can be noted, after activities of the picked event have been rescheduled to different positions, the makespan of the schedule changed from 47 to 45.

Event crossover. The proposed event crossover method operates on the EL representation. Instead of using random activity chains of solution parts to be recombined (i.e. the approach followed in AL-based methodologies), the proposed operator treats events as the solutions' elements for recombination.

From the example project in Figure 4.2 and its EL representation in Figure 4.18 it can be observed that the presented EL consists of four events, each of which contains more than one activity. These events correspond to the peaks of the highest activity, when multiple activities are being started at the same time. At that time, more resources are needed and these periods are considered as periods of high productivity. For the crossover process, such periods in a schedule serve as a desirable characteristic that can be transferred to the offsprings.

Therefore, given two solutions, Parent 1 and Parent 2, the event crossover operator generates an offspring, in such way, that events with the highest amount of activities are inherited from Parent 1, whereas the positions and order of the remaining activities is determined by Parent 2. The algorithmic procedure of event crossover is summarised in Figure 4.22.

Event Crossover

Initialise 2 event lists, E_x and E_y, each consisting of n activities

Sort all events e_k in E_x by size in descending order
Start picking largest e_k from E_x until total amount of picked activities is >= n/2
Pick remaining activities from E_y in the same order as they appear
Form new E_z from picked events
Apply SGS for objective evaluation

Figure 4.22 - Event Crossover pseudo-code

Assume Parent 1 and Parent 2, presented in part a) and b) of Figure 4.23**Error! Reference source not found.**, are two EL representations in the current population that are going to be used to generate a new offspring. The makespan of Parent 1 is 47, while makespan of Parent 2 is 62.

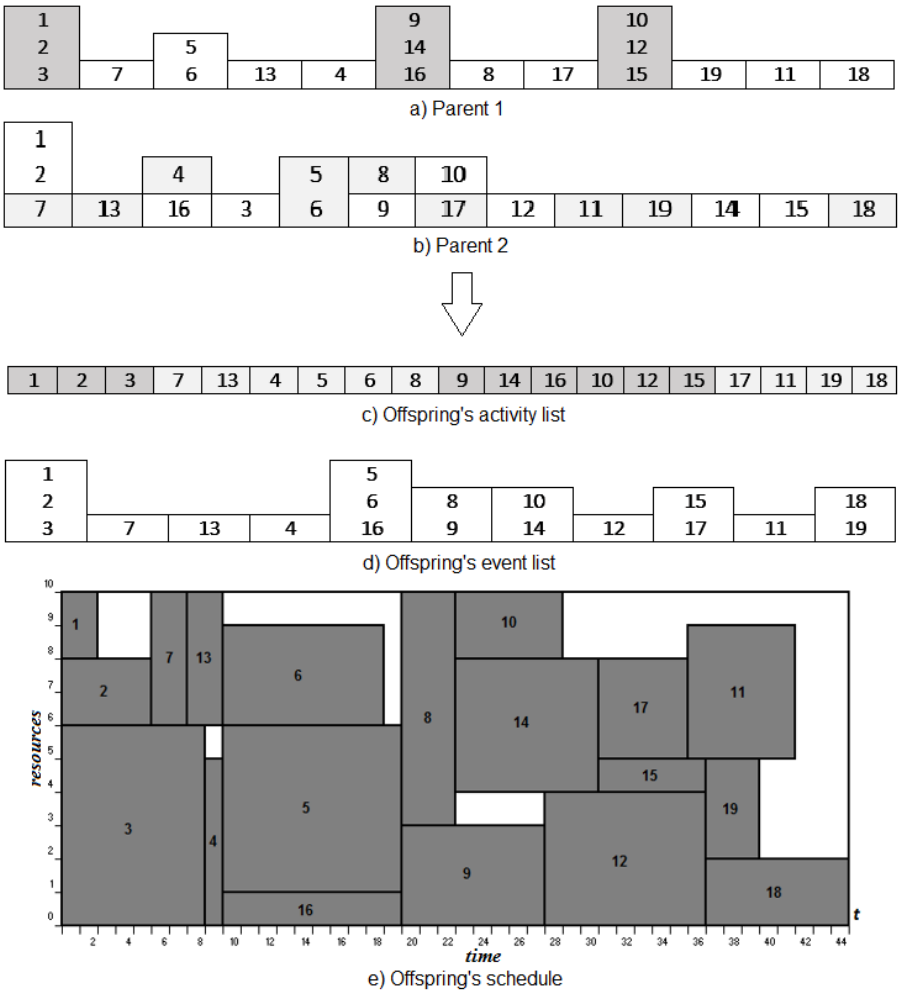

Figure 4.23 - Event Crossover example

The event crossover process begins by ordering events in Parent 1 in descending order by their sizes. Then the events are picked one by one from the largest to the smallest until the number of activities that comprise the picked events is 50% from the total number of activities. For Parent 1, three events are selected for the crossover, each of which consists of three activities. The remaining activities are then taken from Parent 2 in their respective order.

The procedure shown in Figure 4.23 builds an offspring and assigns activities to their respective positions one by one in a growing position order. At each moment, an activity is a candidate if it does not belong to any existing event and has not been assigned, while all its predecessor have already been added. It is worth noting that during its creation, the offspring is represented in a form of AL. The resulting AL, shown in Figure 4.23c, with the application of serial SGS is then converted into EL (Figure 4.23d) by scheduling each activity. Finally, a schedule with the makespan of 45 is produced, as demonstrated in Figure 4.23e.

4.3.1.3 Local Search

Ouaarab et al. (2013) proposed to further improve the original paradigm of CS by adding additional means of creation solutions. In their improvement of the basic CS, the authors added additional mechanism that during each iteration randomly selects fraction of solutions and improves them via local search. The amount of solutions selected for improvement is controlled via smart cuckoo parameter p_c. The name of this parameter is based on the analogy on some cuckoos that are capable of engaging a form of surveillance on an area around nests that are likely to become a host (Payne & Sorenson, 2005).

With the inclusion of the aforementioned mechanism, each turn the improved CS now can create new solutions as follows:

1. Generation of new solution via Lévy flight;
2. A fraction p_c of solutions that are improved via local search
3. A fraction p_a of randomly created solutions

The added process can be illustrated as follows. Suppose that the value of p_c is set to 0.5, then 50% of randomly selected solutions from the population will be modified via one application of event move operator.

4.3.2 Computational Performance

Similarly to previous algorithms in this chapter, performance of IDCS is going to be evaluated by running each of the benchmark instances from PSPLIB. The results of evaluation are then compared with other state-of-the-art heuristic for RSPCP from (Hartmann & Kolisch, 2000) and (Kolisch & Hartmann, 2006). Description of the full experimental setup for parameters tuning and performance evaluation can be found in Section 4.1.3.1.

4.3.2.1 Parameters Settings

The IDCS has four configurable parameters:

- Population size m
- Abandonment rate p_a
- Max amount of steps s
- Portion of smart cuckoos p_c

In order to find the optimal values for these parameters, a sensitivity analysis has been carried out using the *irace* package. The ranges of parameters' values selected for the analysis are summarised **Error! Reference source not found.**.

Table 4.11 - IDCS parameter values for sensitivity analysis

Parameter	Values Range
Population size (m)	[10, 200]
Abandonment rate (p_a)	[0, 0.9]
Max amount of steps (s)	[1, 10]
Portion of smart cuckoos (p_c)	[0, 0.9]

As the result of the algorithm tuning, the optimal parameters values identified by *irace* are summarised in Table 4.12.

Table 4.12 - IDCS optimal parameters values

Parameter	Value
Population size (m)	18
Abandonment rate (p_a)	0.7
Max amount of steps (s)	4
Portion of smart cuckoos (p_c)	0.2

During the tuning process, *irace* iteratively updated the sampling models of the parameters to focus on the best regions of the parameter search space. The frequency of the sampling of parameters values in the regions of the specified parameters search space for m, p_a, s and p_c is presented in Figure 4.24, Figure 4.25, Figure 4.26 and Figure 4.27, respectively.

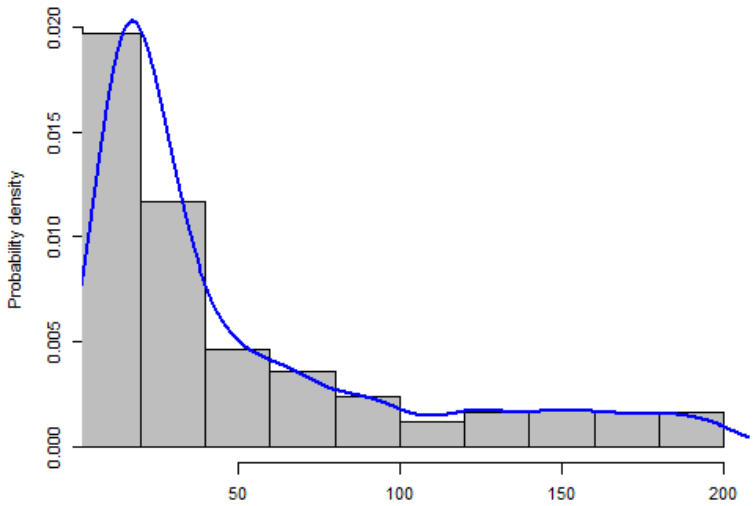

Figure 4.24 - Population size sampling frequency

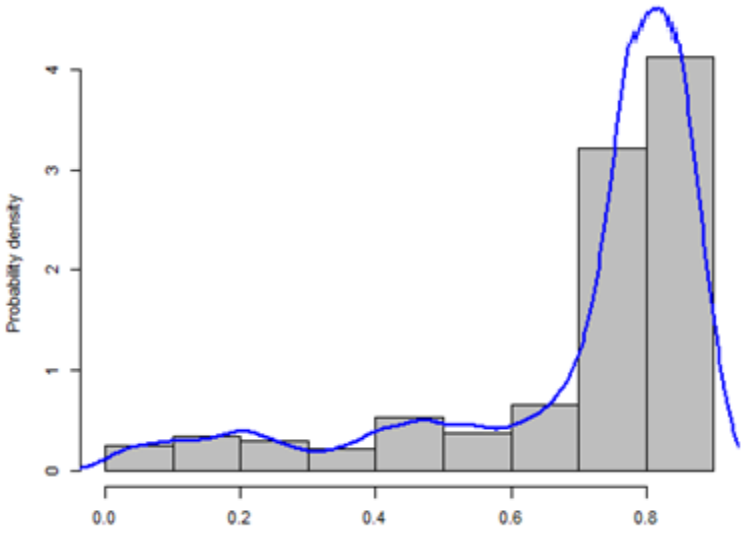

Figure 4.25 - Abandonment rate sampling frequency

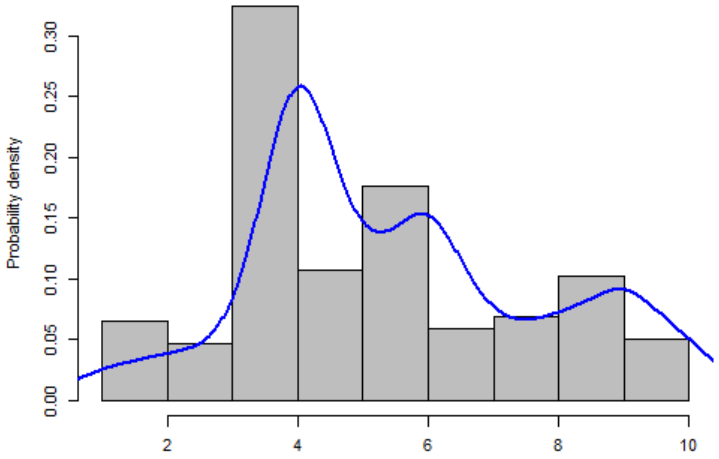

Figure 4.26 - Max amount of steps sampling frequency

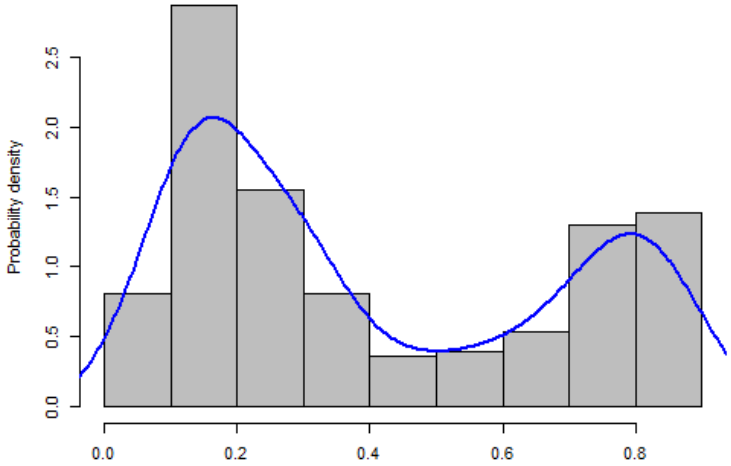

Figure 4.27 - Portion of smart cuckoos sampling frequency

As can be noted from the above-presented tuning results, with the utilisation of more efficient solution representation scheme, the use of new solution generation operators and addition of new mechanics for solution improvement, in

comparison with DCS, the levels of optimal parameters for IDCS almost have not changed. While values for population size and max amount of steps parameters remain the same, the value of abandonment rate parameter reduced from 0.8 to 0.7. Further, Figure 4.28 displays the interaction between parameters and their dependencies on one another on the example of 100 best parameters configurations obtained by *irace* package during fine-tuning.

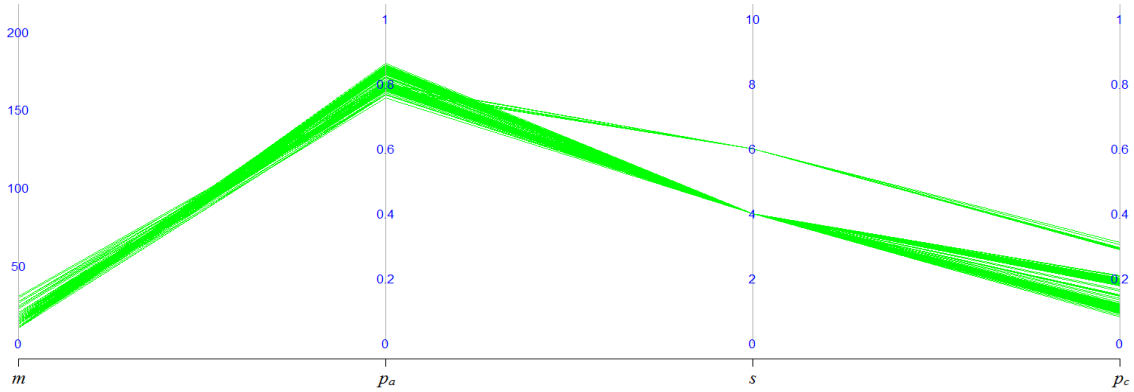

Figure 4.28 - IDCS parameters correlations

From the above-presented results, it is possible to tell that addition of new configurable parameter (p_c) had very little effect on the optimal values of the algorithm's parameters. In comparison to DCS, total variability of optimal parameter combinations has increased, however parameter values of these configurations remained in the same regions:

- m – [10; 30]
- p_a – [0.7; 0.9]
- s – [3; 4]
- p_c – [0.1; 0.3]

4.3.2.2 Comparative Analysis

The computational results of the performance evaluation of IDCS are presented in Table 4.13, Table 4.14 and Table 4.15 for J30, J60 and J120 instance sets, respectively. The first column "Algorithm" reports abbreviations of the algorithms considered for comparison. Column "Author(s)" reports the name(s) of the original author(s) and reference to the work in which the algorithm at hand was previewed. The last column refers to the average deviation % for three stopping conditions: 1000, 5000 and 50000 objective evaluations. Computational performance of other presented algorithms was taken from (Hartmann & Kolisch, 2000) and (Kolisch & Hartmann, 2006).

Table 4.13 - IDCS performance comparison for J30 set

Algorithm	Author(s)	Dev (%)		
		1000	5000	50000
SAILS	Paraskevopoulos et al. (2012)	0.03	0.01	0.00
GA, TS-PR	Kochetov and Stolyar (2003)	0.10	0.04	0.00
SS-PR	Mahdi-Mobini et al. (2009)	0.05	0.02	0.01
GAPS	Mendes et al. (2009)	0.06	0.02	0.01
IDCS	Bibiks et al.	0.09	0.04	0.01
ACOSS	Chen et al. (2010)	0.14	0.06	0.01
SS-FBI	Debels et al. (2006)	0.27	0.11	0.01
GA	Debels and Vanhoucke (2005)	0.15	0.04	0.02
GA-hybrid FBI	Valls et al. (2003)	0.27	0.06	0.02
TS	Nonobe and Ibaraki (2002)	0.46	0.16	0.05
GA	Hartmann (1998)	0.38	0.22	0.08
Sampling + BF	Tormos and Lova (2001)	0.30	0.17	0.09
ANGEL	Tseng and Chen (2006)	0.22	0.09	n/a

Table 4.14 - IDCS performance comparison for J60 set

Algorithm	Author(s)	Dev (%)		
		1000	5000	50000
SAILS	Paraskevopoulos et al. (2012)	11.05	10.72	10.54
SS-PR	Mahdi-Mobini et al. (2009)	11.12	10.74	10.57
GAPS	Mendes et al. (2009)	11.72	11.04	10.67
IDCS	Bibiks et al.	11.78	10.99	10.67
ACOSS	Chen et al. (2010)	11.72	10.98	10.67
GA	Debels and Vanhoucke (2005)	11.45	10.95	10.68
SS-FBI	Debels et al. (2006)	11.73	11.10	10.71
GA-hybrid FBI	Valls et al. (2003)	11.56	11.10	10.73
GA, TS-PR	Kochetov and Stolyar (2003)	11.71	11.17	10.74
GA	Hartmann (1998)	12.21	11.70	11.21
Sampling + BF	Tormos and Lova (2001)	11.88	11.62	11.36
ANGEL	Tseng and Chen (2006)	11.94	11.27	n/a
TS	Nonobe and Ibaraki (2002)	12.97	12.18	11.58

Table 4.15 - IDCS performance comparison for J120 set

Algorithm	Author(s)	Dev (%)		
		1000	5000	50000
IDCS	Bibiks et al.	33.87	32.74	30.47
ACOSS	Chen et al. (2010)	35.19	32.48	30.56
SAILS	Paraskevopoulos et al. (2012)	33.32	32.12	30.78
GA	Debels and Vanhoucke (2005)	34.19	32.34	30.82
GA-hybrid FBI	Valls et al. (2003)	34.07	32.54	31.24
GAPS	Mendes et al. (2009)	35.87	33.03	31.44
SS-PR	Mahdi-Mobini et al. (2009)	34.51	32.61	31.37
SS-FBI	Debels et al. (2006)	35.22	33.10	31.57
GA, TS-PR	Kochetov and Stolyar (2003)	34.74	33.36	32.06
GA	Hartmann (1998)	37.19	35.39	33.21
Sampling + BF	Tormos and Lova (2001)	36.24	35.56	34.77
ANGEL	Tseng and Chen (2006)	36.39	34.49	n/a
TS	Nonobe and Ibaraki (2002)	40.86	37.88	35.85

The results indicate that the proposed IDCS shows itself as a competitive algorithm and performs better or on the same level than more advanced methodologies. IDCS produces the best results regarding the 1000, 5000 and 50000 performance modes for J120 problem set. Considering the 50000 schedules DCS remains one of the most efficient algorithms for solving RCPSP achieving a 30.78% deviation from CP lower bounds.

4.4 Discrete Species Conserving Cuckoo Search

The IDCS presented in the previous section, demonstrated a satisfactory level of performance, comparable to other state-of-the-art methodologies for RCPSPs. However, despite its satisfactory performance, the algorithm cannot be applied to solve the optimisation model formulated in Chapter 3 as it can only obtain one solution candidate at a time. To address this issue, the discrete species conserving cuckoo search (DSCCS) is developed.

The presented DSCCS relies on the species conservation (SC) technique in obtaining multiple global solutions. SC technique, added to the IDCS, is a method of evolving parallel sub-populations. First introduced by Li et al. (2002), this

technique is based on distributed elitism, achieved by identifying in each generation a set of prime individuals that are considered to be worth preserving into the next generation. By running tests on various multimodal optimisation problems from the literature, Li et al. (2002) were able to demonstrate that with the application of SC the algorithm is able to reliably find all possible optimal solutions for all problems under test, as well as achieve competitive performance. The amount of examples of integration of SC into other metaheuristics (Parrot & Li, 2004; Li X.-D. , 2004; Ando, Sakuma, & Kobayashi, 2005; Iwamatsu, 2006; Dong, He, Huang, & Hou, 2005; Stoean, Preuss, Stoean, & Dumitrescu, 2010; Shibasaka, Hara, Ichimura, & Takahama, 2007) only confirms its effectiveness and competitiveness over other multimodal techniques.

The main goal of integration of the SC technique into the IDCS is to divide the whole population into several smaller sub-populations (i.e. species), and each sub-population consists of solutions with similar characteristics. The formation of sub-populations allows the search space to be divided into multiple smaller regions, where each sub-population is responsible for searching for solutions within their specified region. This creates an opportunity for a finer search for a local best optima and provides higher chances of finding global optima. Moreover, multiple solution candidates can be obtained, making the algorithm suitable for application in multimodal scenarios.

4.4.1 Discrete Species Conserving Cuckoo Search for RCPSPs

DSCCS is the result of the integration of SC technique to the original paradigm of the IDCS. As an extension to the IDCS, SC technique is based on the species concept. Species can be defined as a group of individuals (solutions) in a population with similar characteristics that are grouped around individuals with the highest fitness called the species seeds. In the context of the IDCS, the whole population is divided into various sub-populations. Each sub-population consists of solutions that belong to the same species and is centred on the species seed. After all species seeds have been defined and species have been formed, each species seed moves towards a new solution using Lévy flights.

The DSCCS is based on the structure of IDCS and its pseudo-code is shown in Figure 4.29. Changes to the original structure of the algorithm are highlighted in bold.

Discrete Species Conserving Cuckoo Search

Initialise a population P of m host nests $\mathbf{x}_i$, $P_m = (x_1, x_1, \ldots, x_m)$
Set species distance σ_s as the average distance between all $\mathbf{x}_i$

For all $\mathbf{x}_i$ **do**
  Calculate fitness $F_i = f(\mathbf{x}_i)$
End for
While (ObjectiveEvaluationNumber < MaxEvaluationNumber)
  Identify species seeds $\mathbf{X}_s$
  For all $\mathbf{x}_i$ in **$\mathbf{X}_s$ do**
    Create individual ($\mathbf{x}_j$) via Lévy flight
    Calculate fitness $F_j = f(\mathbf{x}_j)$
    Choose random individual $\mathbf{x}_i$ from population P_m
    If ($F_j > F_i$) **then**
      Replace $\mathbf{x}_i$ with $\mathbf{x}_j$
    End if
  End for
  Improve fraction p_c of individuals via local search
  Conserve species seeds $\mathbf{X}_s$
  Abandon a fraction p_a of individuals with worst fitness
  Generate new random individuals
End while
Return species seeds with the highest fitness

Figure 4.29 - DSCCS pseudo-code

The most significant changes to the IDCS are the following:

- After a population initialisation, a species distance parameter σ_s is set
- Within the generation loop, a set $\boldsymbol{X_s}$ of species seeds is determined
- In contrast to the IDCS, where Lévy flights were applied only to the best individual, in the DSCCS the Lévy flights are performed on each species seed
- After execution of a Lévy flights and local search procedures, the species conservation process is performed
- After the operation of the algorithm has ended, multiple solution candidates are identified

4.4.1.1 Definition of Species

Species in the DSCCS are represented by a group of solutions from population that have a set of common characteristics with each other. Goldberg and Richardson (1987) proposed to divide the population into smaller sub-populations

based on the similarity of its members. To estimate the similarity between individuals, they used the Euclidean distance.

The distance between two members of the population $x_i = [x_1, x_2, x_3, \ldots, x_n]$ and $x_j = [x_1, x_2, x_3, \ldots, x_n]$ was estimated as:

$$d_e(x_i, x_j) = \sqrt{\sum_{k=1}^{n} (x_{ik} - x_{jk})^2} \tag{26}$$

The Euclidian distance represents just one of the possible ways of estimating the dissimilarity between two solutions. Moreover, depending on the problem and its setting, different kinds of metrics of distance calculations, hence dissimilarity, need to be applied. Czogalla and Fink (2009) in their analysis of the RCPSP fitness landscape reviewed various distance measure techniques. The reviewed techniques were derived from the interpretation of the permutation representation. In these techniques, the distance between two solutions was calculated in relation to the elements of the permutation, the relative order of the elements, or the absolute position of the elements.

The adjacency distance (Marti, Laguna, & Campos, 2005) is defined as the number of times a pair of activities is adjacent to both solutions x_i and x_j:

$$d_a(x_i, x_j) = \sum_{k=1}^{n} z_k \; ; \; z_k = \begin{cases} 1, & x_{ik} = x_{jk} \; and \; x_{ik+1} = x_{jk+1} \\ 0, & otherwise \end{cases} \tag{27}$$

The precedence distance (Jones, 1995) is the number of times n_{pre} one activity is preceded by another activity in both solutions x_i and x_j:

$$d_p(x_i, x_j) = \frac{n(n-1)}{2} - n_{pre} \tag{28}$$

The absolute position distance (Ronald, 1998) is the number of exact position matches of activities in both solutions x_i and x_j:

$$d_{ap}(x_i, x_j) = n - \sum_{k=1}^{n} z_k \quad with \; z_k = \begin{cases} 1, & x_{ik} = x_{jk} \\ 0, & otherwise \end{cases} \tag{29}$$

The deviation distance (Reeves, 1999) is the difference between starting times $\boldsymbol{S}$ of the activities in both solutions x_i and x_j:

$$d_d(x_i, x_j) = \sum_{k=1}^{n} | S_k^i - S_k^j | \tag{30}$$

In order to analyse applicability of the above-mentioned similarity measures for the RCPSP, Czogalla and Fink (2009) conducted a series of experiments on the sets of benchmark instances from the PSPLIB. As the conclusion of their analysis, the authors noted that algorithms that operated on the deviation distance measure tend to produce better results. Moreover, Chen et al. (2010) and Paraskevopoulos et al. (2012) used the abovementioned distance measure in their algorithms, which confirms its applicability and effectiveness.

In the DSCCS the species are defined in accordance with the species distance parameter σ_s, which signifies the upper bound on the distance between two solutions (i.e. individuals in the population). If a distance $d(x_i, x_j)$ is lower than σ_s, then solutions x_i and x_j are considered to be belonging to the same species. Moreover, σ_s is also used to determine solutions that are going to be preserved into the next generation.

Species are formed from all members of a population $P_m = \{x_1, x_2, x_3, \ldots, x_m\}$. This implies that species are comprised of actual solutions and by no means is just a region of search space.

In DSCCS species are denoted by a set $\boldsymbol{S_i}$. The $\boldsymbol{S_i}$ consists of solutions from population P_m distance between which is less than the σ_s. The species $\boldsymbol{S_i}$ is formed around prime individuals that are called dominating individuals (or species seeds) and denoted as $\boldsymbol{x^* \in S_i}$.

A solution x^* is called dominating if for every other solution $x \in S_i$ in its species

$$f(x^*) \geq f(x) \tag{31}$$

The equality in (31) signifies that one species might be dominated by several individuals.

A species $\boldsymbol{S_i}$ is constructed around a dominating individual $\boldsymbol{x^*}$ if, for every other individual $\boldsymbol{x \in S_i}$,

$$d(x^*, x) \leq \sigma_s / 2 \tag{32}$$

It is worth mentioning that even though species $\boldsymbol{S_i}$ is constructed around the dominating individual $\boldsymbol{x^* \in S_i}$, this does not mean that all solutions $x \in P_m$ within a radius $\sigma_s / 2$ of $\boldsymbol{x^*}$ will be appointed to the same species. An illustration presented in Figure 4.30 shows an example of a distribution of species. Here, members of the population are partitioned into four species. Moreover, it can be noted that as the result of such partition, some solutions were allocated to more than one

species (e.g. solutions in species *1, 2* and *3* in the example). Hence, it can be concluded that several species may be dominated by one dominating individuals, as well as one individual can belong to several species.

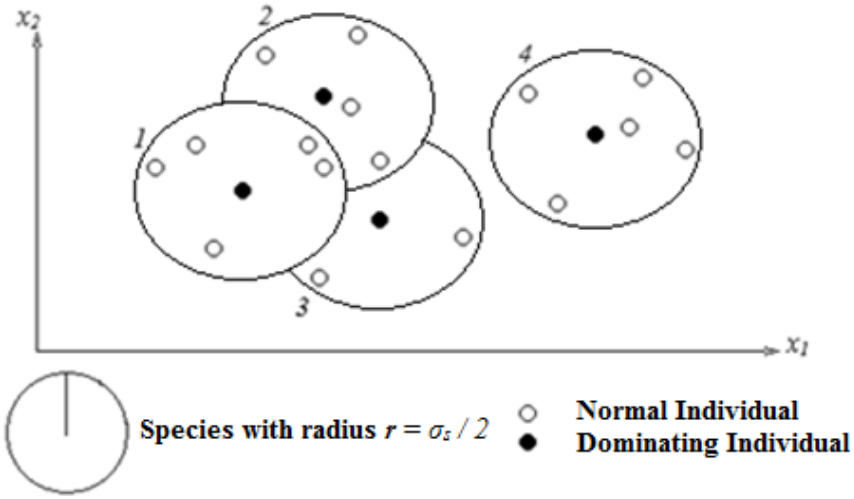

Figure 4.30 - Example of species distribution in a two-dimensional domain

4.4.1.2 Setting of Species Distance

The species distance parameter σ_s plays the most important role in the formation of species and identification of species seeds. If the value of the parameter is set too small, this will result in the formation of too many species and will increase the computational overhead, thus reducing the efficiency of the algorithm. Similarly, a too large value of σ_s will result in too few species being identified.

Deb and Goldberg (1989) in their work implied that each species is enclosed by an n-dimensional hypersphere with radius σ_s. Based on this implication, the authors proposed to estimate the species distance parameter as

$$\sigma_s = \frac{r}{\sqrt[n]{N_g}} \tag{33}$$

where N_g is the known number of global optima and r is the radius of smaller hypersphere containing a feasible space, which can be calculated as

$$r = \frac{1}{2}\sqrt{\sum_{k=1}^{n}(x_k^u - x_k^l)^2} \tag{34}$$

where $x_k{}^u$ and $x_k{}^l$ represent the upper and lower bounds, respectively.

As can be noted from (34), the presented approach will only work if the following conditions are satisfied:

- the number of global optima needs to be known;
- all global optima need to be evenly distributed over the feasible search space.

In most practical applications satisfaction of these conditions is impossible for obvious reasons. To overcome this limitation, Li et al. (2002) proposed to select σ_s such that if the distance between newly found solutions that are significantly different compared to each other is d, then σ_s need to be $\sigma_s \geq 2d$. This approach will guarantee that all members of the population are sufficiently diverse and selected species seeds are adequately different.

Further, in the literature, a variety of techniques with adaptive niche radius has been proposed. The most prominent examples are dynamic fitness sharing (Cioppa, Stefano, & Marcelli, 2007), spatially-structured clearing (Dick, 2010), dynamic niche clustering (Gan & Warwick, 2001), and adaptive co-evolutionary sharing (Goldberg & Wang, 1998). Nevertheless, despite the variety of such techniques, all of them are applied to problems with continuous optimisation domain; therefore, their methods for automatic identification of the niching radius are not applicable to problems with discrete domain. As the compromise to this situation, in DSCCS the species distance parameter is set during the initialisation phase as the average distance value between all members of the initial population.

4.4.1.3 Species Determination

In order to split the population into species and determine which individuals will survive into the next generation, a set of dominating species needs to be identified from the current population. The species determination algorithm developed by Li et al. (2002), shown in Figure 4.31, demonstrates how this is accomplished in the DSCCS. In the presented algorithm, $\boldsymbol{X_S}$ denotes a set of species seeds found in generation $G(t)$.

Species Seeds Determination Algorithm

$\mathbf{X_S} = 0$
While (no more unmarked individuals in $G(t)$)
  Search for the best unmarked $\boldsymbol{x}^* \in G(t)$
  Mark $\boldsymbol{x}^*$ as processed
  found = FALSE
  For all $\boldsymbol{x} \in \boldsymbol{X_S}$ **do**
    If ($d(\boldsymbol{x}^*, x) \leq \sigma_s/2$) **then**
      found = TRUE
      Break
    End if
  End for
  If (found==FALSE) **then**

Figure 4.31 - Species Seeds Determination Algorithm pseudo-code

The algorithm presented in Figure 4.31 builds the set $\boldsymbol{X_S}$ by going through each individual in the current generation $G(t)$ and checking its fitness value against the species seeds found so far. All individuals in $G(t)$ are ordered in decreasing order of their fitness, hence the algorithm starts its way from the fittest individual. If during the comparison $\boldsymbol{X_S}$ does not contain any species seeds that are closer than half the species distance ($\sigma_s / 2$) to the considered solution, then this solution will be added to $\boldsymbol{X_S}$.

The procedure outlined in Figure 4.31 is performed for every generation, and, as result, it creates additional computational overhead. The computational complexity of this operation can be characterised in terms of the number of distance evaluations performed when determining the species seeds in one generation $T_s(m)$ and can be summarised by the following relation:

$$m \leq T_S(m) \leq \sum_{i=1}^{m}(i-1) = \frac{m(m-1)}{2} \tag{35}$$

Therefore, the number of distance evaluations performed for each generation is $O(N^2)$.

4.4.1.4 Species Conservation

Once all species have been identified, a new population is built by applying the usual IDCS operators. Some species may not survive the outcomes of these operations, therefore, they need to be copied (conserved) into the next generation, thus prolonging their existence and maintaining the diversity of a

population. The SC process, developed by Li et al. (2002), is presented in Figure 4.32.

Species Conservation Algorithm

```
Mark all individuals as unprocessed
For all x ∈ X_S do
   Select the worst unmarked y ∈ S'(x, σ_s)
   If (y exists) then
      If (f(y) < f(x)) then
         y = x
      End if
   Else
      Select the worst unmarked y ∈ G(t+1)
      y = x
   End if
End for
```

Figure 4.32 - Species Conservation Algorithm pseudo-code

The SC process shown in Figure 4.32 works as follows:

- New generation $G(t+1)$ is searched for solutions that belong to the same species ($S(x, \sigma_s)$) that have been identified in generation $G(t)$; i.e. individual $y \in G(t+1)$ for which $d(x,y) < \sigma_s /2$
- If species seed $\boldsymbol{x^*}$ is better than the worst of these "similar" solutions, it will replace it
- If $\boldsymbol{x^*}$ is the only member of its species in $G(t+1)$, $\boldsymbol{x^*}$ replaces the worst unmarked solution in $G(t+1)$
- The species seeds are always carried from the previous generation, hence the total amount of species seeds N_S is always smaller than the population size m, and unmarked solutions will always exist

All species seeds are either conserved or replaced by superior individuals of the same species. It is worth mentioning that a scenario when none of the species will be selected for the conservation into the next generation is also possible. However, such a scenario can only happen if the new generation is created by applying operators that contain at least one individual from each of the species defined by the seeds in $\boldsymbol{X_S}$, and if each of these individuals has higher fitness than the corresponding species seed.

SC adds an additional computational overhead. The complexity of this process can be characterised by the amount of distance computations performed when conserving species seeds in one generation $T_c(m)$:

$$m \le T_C(m) \le \sum_{i=1}^{m_S}(m-i+1) = m_S[m-\frac{1}{2}(m_S-1)] < m_S * m \le \overline{m}_\sigma * m \quad (36)$$

Differentiating the above expression for $T_c(m)$ with respect to m_s, it is easily shown that $T_c(m)$ is maximised when $m_s = m + 0.5$. In practice, m_s must be an integer, and, as the species seeds are identified from the population, cannot exceed m. Hence, substituting $m_s = m$ in (35) leads to:

$$T_C(m) \le \frac{1}{2}m(m+1) \quad (37)$$

Combining the results in (31) and (33), the total computational overhead introduced by species conservation technique, as measured by the number of distance calculations performed per each generation, $T_{sc}(m) = T_s(m) + T_c(m)$ is:

$$2m \le T_{SC}(m) \le 2m_S * m \le 2\overline{m}_\sigma * m \quad (38)$$

Thus, the complexity of the number of distance computations performed for each generation is between $O(N)$ and $O(N^2)$. If the species distance σ_s is set small, this results in greater amount of species in each generation, hence the complexity tends to increase towards $O(N^2)$. On the other hand, if σ_s is sufficiently large, there are few species seeds, thus the complexity tend to decrease towards $O(N)$.

4.4.1.5 Global Optima Identification

When stopping condition is met, the DSCCS terminates its operation. At the end of its operation, the algorithm produces the last set $\boldsymbol{X_S}$ of species seeds, e.g. the best solutions that were sufficiently different from each other. This suggests that $\boldsymbol{X_S}$ should contain the found representative solutions, if any. Unfortunately, $\boldsymbol{X_S}$ may contain both low fitness individuals that were stored because they were sufficiently different from all the other individuals in previous generations and high (but not necessarily equal) fitness individuals. Therefore the only solution would be to select from $\boldsymbol{X_S}$ those individuals that have a *high enough* fitness.

In the presented implementation of the DSCCS, the representative solutions to the problem at hand are considered to be the fittest individuals in $\boldsymbol{X_S}$, which are

all individuals in $\boldsymbol{X_S}$ that have a fitness equal to the fitness f_{max} of the fittest species seed.

4.4.2 Computational Performance

Similarly to previous algorithms in this chapter, performance of DSCCS is going to be evaluated by running each of the benchmark instances from PSPLIB. The results of evaluation are then compared with other state-of-the-art heuristic for RSPCP from (Hartmann & Kolisch, 2000) and (Kolisch & Hartmann, 2006). Description of the full experimental setup for parameters tuning and performance evaluation can be found in Section 4.1.3.1.

4.4.2.1 Parameters Settings

Similarly to IDCS, DSCCS has four configurable parameters:

- Population size m
- Abandonment rate p_a
- Max amount of steps s
- Portion of smart cuckoos p_c

In order to find the optimal values for these parameters, a sensitivity analysis has been carried out using the *irace* package. Ranges of parameters value selected for the analysis are summarised Table 4.16.

Table 4.16 – DSCCS parameter values for sensitivity analysis

Parameter	Values Range
Population size (m)	[10, 200]
Abandonment rate (p_a)	[0, 0.9]
Max amount of steps (s)	[1, 10]
Portion of smart cuckoos (p_c)	[0, 0.9]

As the result of the algorithm tuning, the optimal parameters values identified by *irace* are summarised in Table 4.17.

Table 4.17 – DSCCS optimal parameters values

Parameter	Value
Population size (m)	180
Abandonment rate (p_a)	0.7
Max amount of steps (s)	4
Portion of smart cuckoos (p_c)	0.5

During the tuning process, *irace* iteratively updated the sampling models of the parameters to focus on the best regions of the parameter search space. The frequency of the sampling of parameters values in the regions of the specified parameters search space for m, p_a, s and p_c is presented Figure 4.33, Figure 4.34, Figure 4.35 and Figure 4.36.

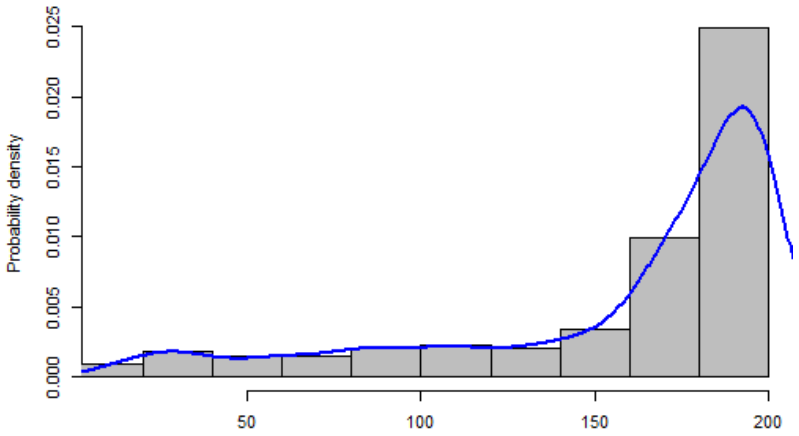

Figure 4.33 - Population size sampling frequency

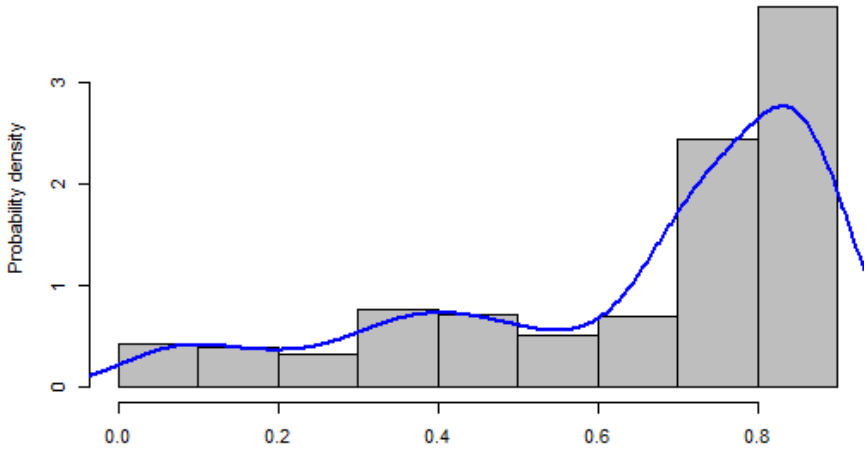

Figure 4.34 - Abandonment rate sampling frequency

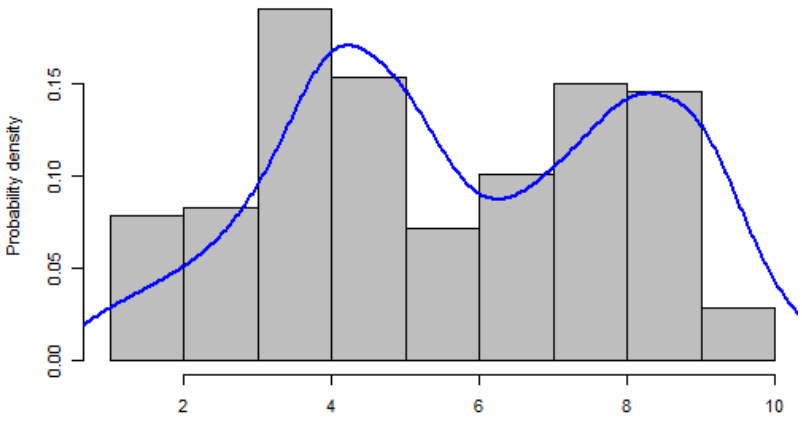

Figure 4.35 - Max amount of steps sampling frequency

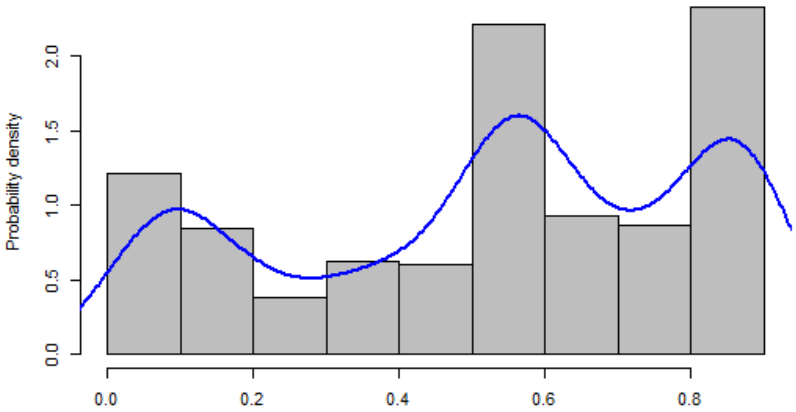

Figure 4.36 - Portion of smart cuckoos sampling frequency

Further, Figure 4.37 displays the interaction between parameters and their dependencies on one another on the example of 100 best parameters configurations obtained by *irace* package during fine-tuning.

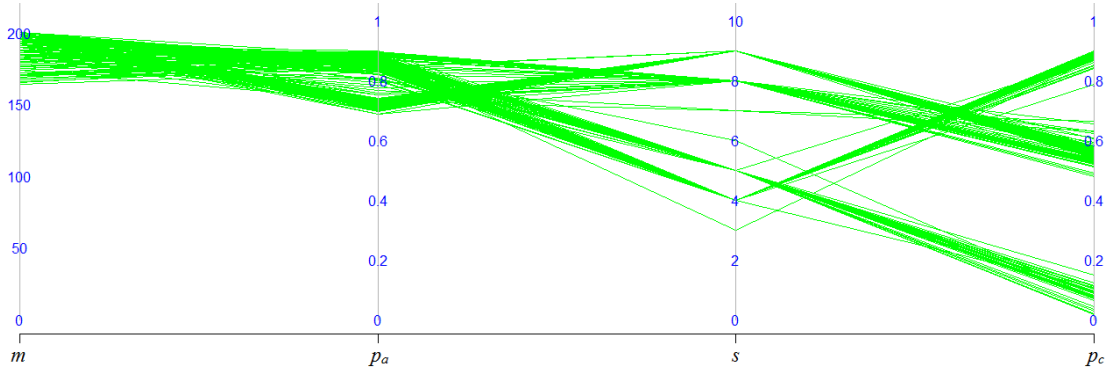

Figure 4.37 – DSCCS parameters correlations

As can be observed from the presented results, with the integration of the species conservation technique into the IDCS, the optimal value for the population size has increased from 18 to 180. This is due to the fact that now for successful operation the algorithm needs to maintain high population of individuals which is divided into species. Having several species allows the algorithm to multiple regions of the solution search space simultaneously, however, in order to do this effectively, high population of diverse individuals is required. With an increase of optimal population size, the optimal value of portion of smart cuckoos parameter p_c increased as well from 0.2 to 0.5. This is explained by the fact that now each turn a set of prime individuals is conserved into the next generation and with the higher p_c there are more chances to improve this individual further.

Moreover, the ambiguity of final best parameter combinations has increased as well. Parameter values of the best configurations were in the following regions:

- m – [160; 200]
- p_a – [0.7; 0.9]
- s – [4; 8]
- p_c – [0.1; 0.9]

4.4.2.2 Comparative Analysis

The computational results of the performance evaluation of DSCCS are presented in Table 4.18, Table 4.19, and

Table 4.20 for J30, J60 and J120 instance sets, respectively. The first column "Algorithm" reports abbreviations of the algorithms considered for comparison. Column "Author(s)" reports the name(s) of the original author(s) and reference to the work in which the algorithm at hand was previewed. The last column refers to the average deviation % for three stopping conditions: 1000, 5000 and 50000 objective evaluations. Computational performance of other presented algorithms was taken from (Hartmann & Kolisch, 2000) and (Kolisch & Hartmann, 2006).

Table 4.18 - DSCCS performance comparison for J30 set

Algorithm	Author(s)	Dev (%)		
		1000	5000	50000
SAILS	Paraskevopoulos et al. (2012)	0.03	0.01	0.00
GA, TS-PR	Kochetov and Stolyar (2003)	0.10	0.04	0.00
SS-PR	Mahdi-Mobini et al. (2009)	0.05	0.02	0.01
DSCCS	Bibiks et al.	0.05	0.02	0.01
GAPS	Mendes et al. (2009)	0.06	0.02	0.01
ACOSS	Chen et al. (2010)	0.14	0.06	0.01
SS-FBI	Debels et al. (2006)	0.27	0.11	0.01
GA	Debels and Vanhoucke (2005)	0.15	0.04	0.02
GA-hybrid FBI	Valls et al. (2003)	0.27	0.06	0.02
TS	Nonobe and Ibaraki (2002)	0.46	0.16	0.05
GA	Hartmann (1998)	0.38	0.22	0.08
Sampling + BF	Tormos and Lova (2001)	0.30	0.17	0.09
ANGEL	Tseng and Chen (2006)	0.22	0.09	n/a

Performance wise, the DSCCS showed the third best result for J30 and J60 sets. For J120 set the DSCCS placed eighth amongst all compared algorithms.

Table 4.19 - DSCCS performance comparison for J60 set

Algorithm	Author(s)	Dev (%)		
		1000	5000	50000
SAILS	Paraskevopoulos et al. (2012)	11.05	10.72	10.54
SS-PR	Mahdi-Mobini et al. (2009)	11.12	10.74	10.57
DSCCS	Bibiks et al.	11.73	11.01	10.62
GAPS	Mendes et al. (2009)	11.72	11.04	10.67
ACOSS	Chen et al. (2010)	11.72	10.98	10.67
GA	Debels and Vanhoucke (2005)	11.45	10.95	10.68
SS-FBI	Debels et al. (2006)	11.73	11.10	10.71
GA-hybrid FBI	Valls et al. (2003)	11.56	11.10	10.73
GA, TS-PR	Kochetov and Stolyar (2003)	11.71	11.17	10.74
GA	Hartmann (1998)	12.21	11.70	11.21
Sampling + BF	Tormos and Lova (2001)	11.88	11.62	11.36
ANGEL	Tseng and Chen (2006)	11.94	11.27	n/a
TS	Nonobe and Ibaraki (2002)	12.97	12.18	11.58

Table 4.20 - DSCCS performance comparison for J120 set

Algorithm	Author(s)	Dev (%)		
		1000	5000	50000
ACOSS	Chen et al. (2010)	35.19	32.48	30.56
SAILS	Paraskevopoulos et al. (2012)	33.32	32.12	30.78
GA	Debels and Vanhoucke (2005)	34.19	32.34	30.82
GA-hybrid FBI	Valls et al. (2003)	34.07	32.54	31.24
GAPS	Mendes et al. (2009)	35.87	33.03	31.44
SS-PR	Mahdi-Mobini et al. (2009)	34.51	32.61	31.37
SS-FBI	Debels et al. (2006)	35.22	33.10	31.57
DSCCS	Bibiks et al.	36.81	33.10	31.96
GA, TS-PR	Kochetov and Stolyar (2003)	34.74	33.36	32.06
GA	Hartmann (1998)	37.19	35.39	33.21
Sampling + BF	Tormos and Lova (2001)	36.24	35.56	34.77
ANGEL	Tseng and Chen (2006)	36.39	34.49	n/a
TS	Nonobe and Ibaraki (2002)	40.86	37.88	35.85

The relatively lower performance in J120 set (in comparison to J30 and J60) can be explained by the fact that benchmark instances with 120 activities have a

lot wider search space, therefore the influence of such parameters as population size m and species distance σs is much higher. In order to provide better results, the algorithm needs to be able to adapt and estimate the optimal values for population size m and species distance σ_s automatically. Nevertheless, these results indicate that DSCCS is capable of finding solutions of high quality with fewer iterations. The DSCCS shows itself as a competitive algorithm and performs better or on the same level than other advanced solutions methodologies.

As the final conclusion, it should be noted that for smaller problem instances with 30 and 60 activities the performance of DSCCS in comparison to the IDSC has improved. However, when solving problems of larger scale, the performance of the algorithm has noticeably degraded. As was stated earlier, this is explained by the fact that the solution search space of problems with larger scale becomes too big, hence setting species distance, as the average distance between all individuals will not work. Therefore, in order to improve the performance, a mechanism for automatic adaptation to the specifics of the problem need to be introduced.

4.4.2.3 Multiple Solutions

One of the features of the DSCCS is the ability to obtain multiple solution candidates for any given RCPSP problem instance. During algorithm testing, depending on the test instance, the amount of found candidate solutions varied from 1 to 22, 1 to 60, and 1 to 21 for J30, J60, and J120 test instances, respectively. It is also worth mentioning that the DSCCS was able to find optimal (or best known) solutions in the majority of all test instances with high rate.

Some feasible solutions for J305_1 test instance are shown in Table 4.21. The optimal makespan (denoted by the starting time of the last activity in the EL) of the test instance is 53. The presented examples indicate that the search space of the RCPSP is indeed filled with a large amount of global optima, as well as prove the capability of DSCCS obtaining multiple solutions.

#	Event list representation																					
1	0										20											
	1						5				22											
	2		4			10	9		11	17	25	23					21					
	3	8	7	12	6	16	15	13	14	18	27	26	30	19	24	29	28	31				
	0	**1**	**5**	**6**	**7**	**11**	**15**	**18**	**22**	**23**	**27**	**30**	**36**	**37**	**40**	**45**	**50**	**53**				
2	0																					
	1																					
	2		6			10		5		17	11	20			23	19		21				
	3	8	7	12	4	16	9	15	13	18	14	22	25	27	26	30	24	29	28	31		
	0	**1**	**5**	**6**	**7**	**11**	**13**	**17**	**18**	**21**	**24**	**27**	**28**	**29**	**30**	**38**	**40**	**46**	**50**	**53**		
3	0																					
	1																					
	2	4		8	6	5				11	20	17	23		25		21					
	3	7	16	12	10	15	13	13	9	14	22	18	27	19	26	24	30	29	28	31		
	0	**5**	**6**	**10**	**11**	**15**	**18**	**18**	**20**	**22**	**27**	**28**	**30**	**34**	**39**	**40**	**42**	**47**	**50**	**53**		
4	0																					
	1						5				20											
	2		4			10	9		11	17	22	19	26	21								
	3	8	7	12	6	16	15	13	14	18	27	23	30	25	24	29	28	31				
	0	**1**	**5**	**6**	**7**	**11**	**15**	**18**	**22**	**23**	**27**	**30**	**36**	**38**	**40**	**44**	**50**	**53**				
5	0																					
	1																					
	2	4		8	6	5			11	20	17	23		25		21						
	3	7	16	12	10	15	13	9	14	22	18	27	19	26	24	30	29	28	31			
	0	**5**	**6**	**10**	**11**	**15**	**18**	**20**	**22**	**27**	**28**	**30**	**34**	**39**	**40**	**42**	**47**	**50**	**53**			
6	0																					
	1														21							
	2	6		4	8	9			17	20	11				25							
	3	7	12	16	10	15	13	5	18	22	14	23	19	27	26	24	30	29	28	31		
	0	**5**	**6**	**7**	**11**	**17**	**18**	**21**	**25**	**27**	**28**	**30**	**31**	**33**	**39**	**40**	**42**	**47**	**50**	**53**		
7	0																					
	1											27										
	2	4		8	6	5			11	20	17	25			24	21						
	3	7	16	12	10	15	13	9	14	20	18	23	26	19	30	29	28	31				
	0	**5**	**6**	**10**	**11**	**15**	**18**	**20**	**22**	**27**	**28**	**30**	**34**	**39**	**40**	**47**	**50**	**53**				
8	0																					
	1								5						21							
	2		6			4			15		20	11	19		25							
	3	8	7	12	16	10	9	13	17	18	22	14	23	27	26	24	30	29	28	31		
	0	**1**	**5**	**6**	**7**	**11**	**13**	**18**	**21**	**24**	**27**	**28**	**30**	**33**	**38**	**40**	**42**	**46**	**50**	**53**		
9	0																					
	1														21							
	2	5		4			8		5		20	11	19		25							
	3	6	12	16	10	9	15	13	17	18	22	14	23	27	26	24	30	29	28	31		
	0	**5**	**6**	**7**	**11**	**13**	**17**	**18**	**21**	**23**	**27**	**28**	**30**	**33**	**38**	**40**	**42**	**46**	**50**	**53**		
10	0																					
	1										17											
	2	4		8	6	5			11		18	20						21				
	3	7	16	12	10	15	13	9	14	22	27	23	25	26	19	24	30	29	28	31		
	0	**5**	**6**	**10**	**11**	**15**	**18**	**20**	**22**	**27**	**28**	**30**	**33**	**34**	**37**	**40**	**42**	**45**	**50**	**53**		

Table 4.21 - Example event list representations of found solutions of J30 test instance

4.5 Summary

In the first two sections of this chapter two novel algorithms, CS and FPA, are modified and applied to solve the RCPSP. The ideas of CS and FPA are extended

to a discrete domain, resulting in the development of DCS and DFPA, respectively. As the original versions of these algorithms were specifically designed for the application in continuous optimisation problems, in order to apply them for solving RCPSPs, the algorithms' key components and elements are reinterpreted. These include solution representation and encoding scheme; exploration and exploitation of the solution search space; and crossover operator in the case of DFPA.

Some preliminary computational experiments were carried out to test the suitability of the algorithms when applied to solve RCPSPs on the sets of benchmark instances from PSPLIB and the results of tests were compared with the performances of other non-hybrid algorithms for RCPSPs from the literature. The comparative analysis showed that both algorithms have a competitive level of performance. Nevertheless, DCA and DFPA have many areas for improvement, which include: inefficient solution representation scheme, use of randomisation-reliant and context-free operators, and the creation of random individuals.

To address and fix the limitations of the DCS and DFPA, the IDCS was proposed. The IDCS represents an improved version of the DCS and it introduces the following changes to the original paradigm of DCS:

- use of the EL as more efficient solution representation scheme;
- addition of a new mechanism for improvement individuals in the current population via local search;
- use of the event move operator that takes into account properties of the activities (successors and predecessors) to improve the current solution; and
- the event crossover which is designed to combine useful problem-specific information extracted from the parent for the purpose of generating high quality children.

The performance of the IDCS was evaluated using all benchmark instances from J30, J60, and J120 sets from PSPLIB and the performance evaluation results are compared against other state-of-the-art algorithms for RCPSPs from literature. The results show that IDCS outperforms most of the chosen state-of-the-art algorithms for the instance sets J30, J60, and J120.

Nevertheless, due to the specifics of the optimisation model proposed in this thesis, one of the areas of improvement of the algorithm is its adaptation for

application in multimodal scenarios. Thus, by combining the IDCS and SC technique, a new metaheuristic algorithm DSCCS was subsequently proposed.

The application of the species conserving technique to the IDCS has made this algorithm suitable for the use in multimodal scenarios. The main difference between the IDCS and DSCCS is the introduction of two new additional operators: species determination and species conservation processes. Moreover, in contrast to the IDCS where Lévy flight is only applied on the fittest individual, in the DSCCS the Lévy flight is applied to all species seeds.

The computational results show that DSCCS is a high-quality algorithm which is capable of achieving high performance comparable to other state-of-the-art heuristics for the RCPSP, as well as obtaining multiple solution candidates.

Chapter 5 Case Studies

This chapter focuses on the application of the DSCCS, presented in Chapter 4, to the HPMP, proposed in Chapter 3. For this, two sets of experiments are conducted: activity scheduling and resource allocation of the real-world project, and the algorithm's performance evaluation on the set of benchmark instances.

For the first experiment, the DSCCS is applied to schedule activities of the real-life software development project. The project consists of 51 different activities with complex precedence relationships and 6 types of resources.

For the second experiment, several of the most popular methodologies for the RCPSP are implemented and then applied to solve the benchmark instances from PSPLIB, which were edited to feature additional parameters for resource efficiency and learnability.

5.1 HARNet Automated Testing System Project

Harmonised Antennas and Radio Networks (HARNet) represents a large-scale aeronautical R&D project and is characterised by a complex structure, reliance on intensive high-tech, a collaboration of many partner companies, and exposure to considerable uncertainties and risks. Because of that, the development of HARNet was split into smaller work packages (WPs) that were distributed among the partners as follows:

- WP1: Identification and management of the project transversal activities
- WP2: Next generation antennas design study and development
- WP3: Next generation communication environment development
- WP4: Design and synthesis of the harmonised amplifier sets
- WP5: Design and synthesis of the harmonised transceiver
- WP6: Design and synthesis of an I/Q baseband bus system
- WP7: Design and development of the baseband processing system
- WP8: Design and development of system capable of providing high burst capacity data transfers for entertainment and system upgrades
- WP9: Design and development of the automated testing environment

The WP1 is a managerial project objective of which is to identify, assess and manage the wide range of general risks and requirements subject to all

development stages of the HARNet. WPs ranging from WP2 to WP8 are hardware related projects, main concern of which is the study, design and development of the certain HARNet elements, which, when combined together, will form the entire system. The last WP in the list, WP9, in comparison to other WPs, represents a software project. Its objective is to produce an overarching automated test system that will cooperate with external and third-party testing mechanisms to carry out the testing of the HARNet components at different stages of the development cycle.

Due to the previous experiences of being involved only with hardware-related projects, planning and scheduling of the design and development stages of the WP9 (HARNet automated testing system (HATS)) were new for the HARNet project managers. The management of such project required more efficient tools and models. Without such tools, the managers had to face the following managerial challenges:

- *Resource allocation and scheduling* – the development cycle of WP9 consisted of set of activities which required collaboration of different teams with various specialities and from different departments
- *Uncertain activity durations* – most of the activities in the WP9 were new for the project managers and developers. This created the difficulty of the evaluation durations of activities
- *Time dependency* – as the development of the project went ahead, related factors might evolve with time, influencing the duration of the activities. For instance, as experience accumulated and technological maturity grown, technology risks tended to decrease
- *Conflicting objectives* – WP9 involved multiple conflicting objectives, such as minimisation of makespan and maximisation of the competency of selected group members

For the development of WP9, focus on the above-mentioned challenges could provide managers with effective candidate solutions to shorten the development cycle, save costs and improve efficiency. This could be achieved by designing efficient optimisation and decision support models.

5.1.1 Project Description

The main outcome of the WP9 was envisioned to be the overarching automated testing system that would aim to cooperate with external and third-party testing

mechanisms to carry out testing and validation of many different radio components. Moreover, because of its generic and extensible design, it could also be used to test other software and hardware components. Support for testing of new components could be easily added due to the extensible nature of the system. Further, HATS was projected to allow interfacing with existing testing tools, so that all testing can be carried out and orchestrated from one overarching system.

5.1.1.1 Project Background

The main content of the WP9 included the design and development of the HATS prototype system and its subsequent integrations and system tests. The overall project's lifetime could be divided into the following stages:

- Stage 1: Review of the testing technologies and identification of the functional requirements
- Stage 2: System architecture design
- Stage 3: Synthesis of the test prototype system
- Stage 4: Testing and verification of the final system

The functional requirements of the HATS prototype were identified in collaboration with partners from other WPs and covered only the bare minimal set of functionalities that needed to be implemented.

Because of the HATS' generality and extensibility, the system architecture had a modular structure and was based on the principles of the service-oriented architecture (SOA) and modular programming.

The functionality of HATS was separated into independent, interchangeable modules, such that each contained everything necessary to execute only one aspect of the desired functionality. As the development process went further, the core functionality of HATS could be expanded by the addition of new modules. Moreover, such approach has also helped to split the project activities equally between all groups of developers, making each group capable to independently develop and test their assigned module.

5.1.1.2 Project Network

The development cycle of HATS consisted of 51 activities, which are listed in Table 5.1,

Table 5.2,

Table 5.3 and

Table 5.4. Project networks presented in Figure 5.1 and Figure 5.2 characterise the precedence relations of all activities that are comprised in the project. Activities A_0 and A_{52} are dummy activities that signify the start and the end of the project, respectively, and have a deterministic duration of 0. The preliminary duration estimations p_i* of all other non-dummy activities A_i can be found in the last column of the respective tables and are expressed in weeks.

Table 5.1 - WP9 Stage 1 activities

A_i	Name	p_i*
A_1	Review of tools, techniques, and methodologies	4
A_2	Review of automated testing technologies	4
A_3	Selection of tools and technologies	2
A_4	Determination of operating environment and preconditions	3
A_5	Determination of HATS components and subcomponents	4
A_6	Definition of testing scopes and interfaces	2
A_7	Definition of functional requirements	6

Table 5.2 - WP9 Stage 2 activities

A_i	Name	p_i*
A_8	Identification of functional components	2
A_9	Identification of system specification	2
A_{10}	Software architecture design	7
A_{11}	Hardware architecture design	6
A_{12}	Graphical user interface (GUI) design	7
A_{13}	Component design	5
A_{14}	Database design	3
A_{15}	Definition of test cases	5
A_{16}	Definition of physical and logical interfaces	3
A_{17}	Definition of test flow diagrams	9

Table 5.3 - WP9 Stage 3 activities

A_i	Name	p_i*
A_{18}	Core engine development	8
A_{19}	Communication manager development	2
A_{20}	User manager development	2
A_{21}	Component manager development	4
A_{22}	Test scenario manager development	6
A_{23}	Test scheduling manager development	10
A_{24}	Test execution manager development	9
A_{25}	Test results manager development	2
A_{26}	Maintenance manager development	2
A_{27}	Adaptation manager development	5
A_{28}	System configuration manager development	3
A_{29}	Utility manager development	2
A_{30}	System under test adapters development	4
A_{31}	GUI core engine development	6
A_{32}	GUI test designer and planner development	6
A_{33}	GUI results processor and reporter development	2
A_{34}	GUI component configuration panel development	3
A_{35}	GUI system configuration panel development	3
A_{36}	GUI user configuration panel development	2

Table 5.4 - WP9 Stage 4 activities

A_i	Name	p_i*
A_{37}	Backend integration testing	6
A_{38}	Frontend integration testing	5
A_{39}	Whole system integration testing	7
A_{40}	Component interfaces testing	5
A_{41}	System performance testing	4
A_{42}	Compatibility testing	3
A_{43}	Scalability testing	3
A_{44}	Maintenance testing	5
A_{45}	GUI testing	5
A_{46}	Usability testing	3
A_{47}	Security testing	3
A_{48}	Regression testing	3
A_{49}	Testing among different stations	2
A_{50}	Joint testing and verification	7
A_{51}	Operational acceptance testing	4

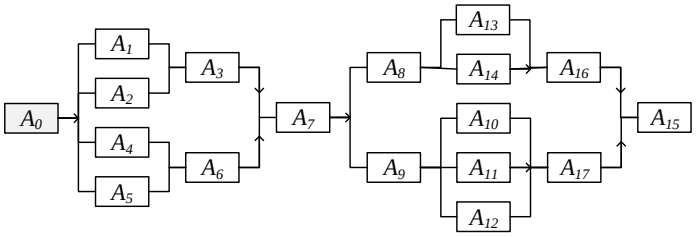

Figure 5.1 – WP9 project network of stages 1 and 2

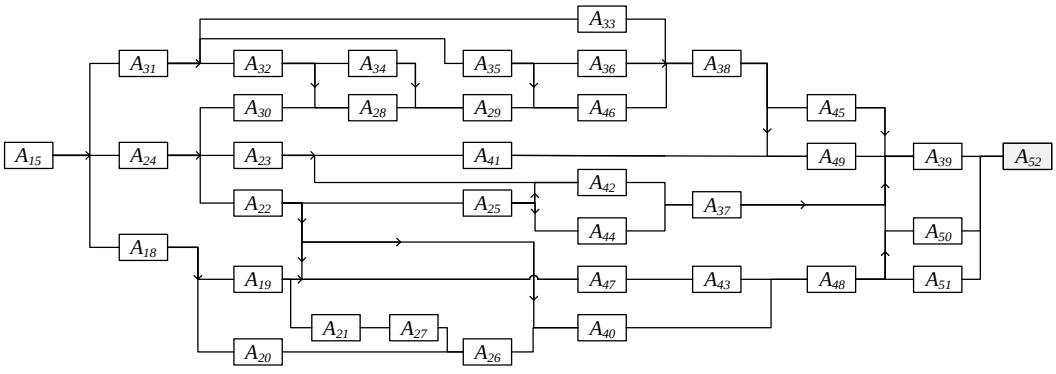

Figure 5.2 – WP9 project network of stages 3 and 4

As can be observed from the project networks presented in Figure 5.1 and Figure 5.2, stages 1 and 2 are relatively sequential and consist of the activities

with easy-to-follow relationships. However, stages 3 and 4 contain activities with complex dependency networks scheduling which is a challenging task.

5.1.1.3 Project Resources

For execution of all planned activities of the WP9 project several types of resources were needed. All resources that are considered in this case study have limited capacities, therefore, inappropriate scheduling and allocation may cause conflicts among the tasks. The total number of resources that are available for completion of the case study and their efficiencies are presented in Table 5.5.

Table 5.5 - WP9 Resource capacities

R_k	Description	RC_k	e_k	l_k
R_1	Researchers in the frontend department	4	0.55	15
R_2	Researchers in the backend department	9	0.40	15
R_3	Researches in the design department	5	0.35	20
R_4	Researchers in the system engineering department	5	0.25	25
R_5	Network analyser	2	n/a	n/a
R_6	Network router	1	n/a	n/a

Resource requirements of each of the case study's activities are shown in Table 5.6,

Table 5.7,

Table 5.8 and

Table 5.9.

Table 5.6 - WP9 Stage 1 resource requirements

A_i	R_1	R_2	R_3	R_4	R_5	R_6
A_1	3	3	1	1	0	0
A_2	1	1	3	3	0	0
A_3	2	2	2	2	0	0
A_4	0	0	2	3	0	0
A_5	1	1	2	2	0	0
A_6	0	0	2	4	0	0
A_7	2	2	1	1	0	0

Table 5.7 - WP9 Stage 2 resource requirements

A_i	R_1	R_2	R_3	R_4	R_5	R_6
A_8	0	0	2	2	0	0
A_9	0	1	1	3	0	0
A_{10}	1	3	3	1	0	0
A_{11}	0	1	2	3	0	0
A_{12}	4	0	2	0	0	0
A_{13}	0	2	2	2	0	0
A_{14}	0	2	2	0	0	0
A_{15}	3	3	0	0	0	0
A_{16}	0	0	2	3	0	0
A_{17}	3	3	0	0	0	0

Table 5.8 - WP9 Stage 3 resource requirements

A_i	R_1	R_2	R_3	R_4	R_5	R_6
A_{18}	0	4	2	1	0	0
A_{19}	0	1	1	3	1	1
A_{20}	0	2	2	0	0	0
A_{21}	0	3	1	2	1	0
A_{22}	0	3	3	1	0	0
A_{23}	0	4	2	0	0	0
A_{24}	0	4	2	2	1	1
A_{25}	1	2	1	0	0	0
A_{26}	0	1	1	3	0	0
A_{27}	0	2	2	3	1	0
A_{28}	1	1	0	3	1	1
A_{29}	0	2	2	1	0	0
A_{30}	0	0	2	4	1	0
A_{31}	4	0	3	0	0	0
A_{32}	3	1	3	0	0	0
A_{33}	3	0	1	1	0	0
A_{34}	2	0	1	2	0	0
A_{35}	2	0	1	3	0	0
A_{36}	3	0	1	0	0	0

Table 5.9 - WP9 Stage 4 resource requirements

A_i	R_1	R_2	R_3	R_4	R_5	R_6
A_{37}	0	4	1	2	0	0
A_{38}	4	0	1	2	0	0
A_{39}	3	3	2	2	2	0
A_{40}	0	2	0	3	1	1
A_{41}	1	3	1	3	0	0
A_{42}	2	0	3	2	0	0
A_{43}	2	0	3	2	0	0
A_{44}	0	0	3	3	0	0
A_{45}	4	0	1	2	0	0
A_{46}	2	1	2	1	0	0
A_{47}	0	1	2	2	0	1
A_{48}	2	2	2	1	0	0
A_{49}	2	2	0	3	2	0
A_{50}	3	3	2	2	1	0
A_{51}	2	2	3	3	0	0

Resources R_1, R_2, R_3, and R_4 represent people, i.e. departments in which each unit of these resources corresponds to one researcher or developer. In this case study, four groups of researchers are considered, corresponding to different areas of expertise and different levels of experience: frontend, backend, design, and system engineering. Depending on the activity's requirement, several units (researchers) of a particular resource type might be needed for its execution. If so, the researchers will work in cooperation as a group and support the design specification, organisation of the interface, review, technical consulting, etc. Consequently, activities are completed by different groups of researchers under a higher-level global plan. Resources R_5 and R_6 represent special equipment needed for completion of some of the activities. In contrast to resources R_1, R_2, R_3, and R_4, resources R_5 and R_6 cannot gain any experience, thus, they cannot influence the duration of activities.

Since the actual data of the WP9 project is confidential, the values e_k and l_k in this case study have been chosen artificially for the purpose of illustrating properties of the proposed optimisation model. In the context of this work, value e_k represents the maximum efficiency gain that can be achieved by resource type

by learning and gaining relevant experience. Value e_1 = 0.25, for example, denotes that a highly experienced group of researchers can be up to 25% more efficient than it was before the project started. The learnability coefficient l_k measures the learning effect through experience: a higher value of this parameter suggests that the experience gain effect will require more time. A simple method to estimate l_k in the real-world project would be via a management report. For example, statement like “90% of the maximum experience gain effect for the resource R_k can be accomplished after 30 weeks of practice” can lead to the following:

$$90\% = \exp(-\frac{l_k}{30}) \tag{39}$$

which yields the value $l_k = 30*(-ln(0.9)) = 3.1608$. The l_k-values selected for this case study in **Error! Reference source not found.** are much higher and they correlate to a rather slow learning effect.

5.1.2 Algorithm Application

Application of any algorithm for the RCPSP-kind of problem mainly focuses on the reinterpretation of its key elements, which include solution representation, objective function, and genetic operators. Additionally, application of techniques for multimodal optimisation would also require reinterpretation of metrics of similarity estimation between members of the population.

To test the validity of the proposed optimisation model and solve the presented case study, the DSCCS is going to be utilised. For more detailed description of the DSCCS refer to Section 4.4. The experimental evaluation of the DSCCS on sets of benchmark instances from PSPLIB for the standard RCPSP demonstrated the capability of the algorithm to confidently handle RCPSP instances of various sizes, ability to find multiple optimal solutions and competitiveness against other state-of-the-art methodologies for the deterministic RCPSP.

As DSCCS was specifically developed for solving the derivatives of RCPSPs, in order to apply it to the proposed optimisation model the only element that needs to be reinterpreted is the objective function. The considered optimisation model considers optimisation of two objectives: makespan minimisation and resource efficiency balancing. In the standard RCPSP, the makespan of a schedule is

estimated by assigning starting time to each of the schedule's activities using serial or parallel schedule generation scheme (SGS). Due to the specifics of the proposed model, the standard variations of the SGS are not applicable here, hence a new scheme needs to be introduced that will take into account the varying durations of the activities. The estimation of resource efficiency balancing will follow the procedure described in Section 3.4.2. The reinterpretation of other algorithm's key elements is not necessary and they will remain as follows:

- Solution representation – event list, presented in Section **Error! Reference source not found.**
- Lévy flight and local search – event move, described in Section 4.3.1.2
- Generation of new individuals – event crossover, proposed in Section 4.3.1.2
- Similarity estimation – deviation distance, described in Section 4.4.1.1

5.1.2.1 Makespan Estimation

In the standard deterministic RCPSP, the schedule's makespan is estimated by converting a solution into a schedule by successively scheduling activities one by one in the same order as they appear in the solution. Once the schedule is created, the makespan is equal to the starting time S_{n+1} of the last dummy activity A_{n+1}. The conversion of a solution into a schedule is accomplished by applying SGS. For the traditional RCPSPs, Kolisch (1996) outlined two types of SGS: serial and parallel.

Serial SGS uses the activity incrimination approach and schedules one activity at a time as early as possible while satisfying precedence and resource constraints. This is achieved by producing the list of activities and selecting at each stage the next activity from the list to schedule it at its first possible starting time without violating both the precedence and resource constraints.

Instead of iterating over the activity list, parallel SGS iterates over the time horizon of the project and schedules the eligible activities. The scheme starts at time point t = 0 and attempts to schedule all activities eligible for scheduling at this time point. Once this is done, the time pointer is increased. At each decision point, the eligible activities are scheduled with a starting time equal to the decision point. Activities that cannot be scheduled due to the resource conflict are skipped and become eligible to schedule at the next decision point.

Due to the properties of these SGSs, the schedules that are generated using serial SGS are called active schedules, meaning their activities are scheduled as early as possible and rescheduling them to earlier starting times will violate the precedence or resource constraints. The schedules that are created by parallel SGS are called *non-delay schedules*. Non-delay schedule is a schedule where no resources were kept idle at a time when it could begin processing an activity. Kolisch (1996) in his experimental evaluation of both scheduling modes showed that in the majority of conducted experiments serial SGS produced better results, mainly due to the inability of parallel SGS to reach an optimal solution in some cases.

The SGS for the proposed optimisation model is based on the serial SGS, primarily because of its superiority over parallel SGS. The pseudo-code of the algorithm used for converting activity sequence $\boldsymbol{A}$ of the EL representation into the schedule and estimating its makespan S_{n+1} is shown in Figure 5.3.

Makespan Estimation Algorithm

```
checkFeasibility(A)
For all A_i ∈ A do
    Find the earliest possible starting time t
    Calculate duration p_i^m for time t
    While(checkSchedulability(A_i, p_i, t) == false)
        t = t + 1
        Calculate duration p_i^m for new starting time t
    End while
    Set S_i = t
    Update resource usage matrix
End for
```

Figure 5.3 – Makespan estimation algorithm pseudo-code

The makespan estimation process shown in Figure 5.3 begins with checking the feasibility of the inputted activity sequence. Two conditions are checked here: the sequence has to start and end with the starting and ending dummy activities, respectively; activities in the sequence need to be placed after their respective predecessors. If one of these conditions is not satisfied, the algorithm will not proceed. Once the feasibility of the sequence is verified, for each subsequent activity in the given sequence the following is performed:

- Taking into consideration the information about activity A_i predecessors, estimate its earliest possible starting time t. In this case, it is equal to

the latest finishing time of one of its predecessors. At this step the resource availabilities are not considered

- Calculate new activity duration $p_i(t)$ at time t taking into account resource experience
- Check resource availabilities for a time period $[t, t + p_i^m]$. If scheduling at this time period is not possible, increment time by 1 and recalculate new duration for a new time. This step is repeated until the feasible time is found
- Set starting time S_i of activity A_i to t and update resource usage matrix for subsequent operations

After all activities have been scheduled, the schedule duration is equal to the starting time of the last dummy activity S_{n+1}.

The procedure of makespan estimation is applied to each new EL after its creation.

5.1.2.2 Resource Efficiency Balancing

In order to provide stable and consistent execution of a project, resource efficiency balance is necessary. If some resources have very high experience while others are lacking it, then the execution of some activities might be necessarily prolonged. Moreover, this might also leave some resources idle for some periods of time.

The majority of traditional RCPSPs only consider makespan minimisation is their primary and the only one objective. For the proposed optimisation model, the resource efficiency balancing is combined with the process of best solution identification which is performed at the end of DSCCS operation. The pseudo-code of this procedure is shown in Figure 5.4. In the context of given problem, resource efficiency balancing takes into account the fairness measure which ensures that all resources are distributed equally and fairly among all activities and receive similar amount of experience.

Resource Efficiency Balancing Algorithm

Find the smallest $f_1(x) \in \boldsymbol{X_s}$; $r_{ref} = f_1(x)$
Filter $\boldsymbol{X_s}$ to include only individuals with makespan of r_{ref}
For all $x \in \boldsymbol{X_s}$ **do**
  Estimate $f_2(x)$
End for

Find the smallest $f_2(x) \in \boldsymbol{X_s}$;

Figure 5.4 – Resource efficiency balancing algorithm

The resource efficiency balancing begins with the search of the species seed from the species seeds set $\boldsymbol{X_s}$ with the lowest makespan. The value of the found species seed is then used as a reference to filter all other species seeds which have higher makespan. As was shown by previous researches and experiments (Czogalla & Fink, 2009; Ikeda & Kobayashi, 2000; Pérez, Posada, & Lorenzana, 2015), RCPSPs have multimodal landscapes which correspond to the existence of multiple global solutions. Therefore, when species seeds set $\boldsymbol{X_s}$ is filtered to contain only individuals with the best makespan, it is expected that multiple individuals will remain. After it is done, for each of the remaining species seeds the resource efficiency balance is estimated. At the end, the algorithm selects the individuals with the best balancing.

The procedure outlined in Figure 5.4 is only performed once as the very last stage of the algorithm's operation.

5.1.3 Experiment Setup and Parameter Choices

As was shown by previous experiments, the algorithm's parameters have a significant impact on its performance, quality of the received solutions, computational time and the success rate. The DSCCS has four parameters that can be configured: population size m, abandonment rate p_a, a portion of smart cuckoos p_c, max amount of steps s and configurable species distance σ_s. The effects of these parameters and their influence on the quality and amount of received solutions are studied in Section **Error! Reference source not found.**. For the proposed case study, the performance is going to be examined by applying different algorithmic settings for all parameters. The values of these parameters are summarised in Table 5.10, where σ_s* represents the average distance between all individuals in the population and is calculated automatically by the algorithm.

Table 5.10 - DSCCS parameter choices for the case study

Parameter	Value
Population size (m)	180
Abandonment rate (p_a)	0.7
Portion of smart cuckoos (p_c)	0.5
Max amount of steps (s)	4

Species distance (σ_s*)	σ_s*
Stopping criterion	[5000, 50000]

Values of the DSCCS parameters for this experiment are going to be set to the values specified in the table above. These values were obtained through automatic fine-tuning of the algorithm via application of *irace* package on benchmark test instances from experimental setup described in Section 4.1.3.1. The main reason for using several combinations of parameters for stopping criterion is to analyse the behaviour of the DSCCS on the presented case study, see what kind of impact its parameters will have when applied to solve a practical example, and identify the minimal value sets required to find the optimal solution for given scenario.

In order to experimentally validate the proposed optimisation model and see how efficiency and learnability coefficients affect the durations of *activities*, two sets of experiments will be carried out: deterministic scheduling, which assumes that activity durations are constant, and stochastic scheduling in which activity durations can change depending on the experience of the applied resources and execution time. Before applying DSCCS to solve the instances of the problem, 10000 randomly-generated schedules are created and their properties are examined. This step is required so later it would be possible to analyse the performance of the algorithm and visualise how effective it was in scheduling the case study. For both experiments, the algorithm is going to be applied 1000 times to solve the presented case study. The results of these are then averaged and presented in the subsequent sections.

In order to avoid the impact of randomness, 100 independent runs for each experiment are carried out and the results of these runs are averaged.

5.1.4 Experimental Analysis

The results presented in this section pertain to the case study presented in Section 5.1.1. Three criteria for the performance evaluation are considered:

- Makespan of the final solution(s);
- Amount of obtained optimal solutions;
- Computational time.

It is worth mentioning that for the amount of obtained optimal solutions criterion solutions are accounted if they are considered to be different enough and their

makespan is equal to the makespan of the reference solution. In this experiment, the reference solution is the best solution that was obtained by the algorithm throughout all experiments. The similarity of the solutions is identified with respect to the value of species distance parameter σ_s via application of deviation distance measure (Reeves, 1999).

5.1.4.1 Deterministic Scheduling

Randomly-generated schedules. Before any assessment of the algorithm's performance can begin, first, it is necessary to identify the durations of randomly generated schedules. This is required as later these durations can be compared with the results received by the algorithm to evaluate its performance and to see whether there are any improvements in the schedules obtained by the algorithm and how effective these improvements are, if any. To identify these values, for statistical purposes four sets of randomly generated feasible deterministic schedules are created. The received data is summarised in Table 5.11, where size denotes the number of random schedules in each set, σ_s* is the average distance between every pair of solutions, min. ms. is the minimal makespan, mean ms. – mean value of the makespan, median ms – median of the makespan, max. ms. – maximum makespan and st. dev. – standard deviation.

Table 5.11 - Durations of randomly-generated deterministic schedules

Size	σ_s*	Min. ms.	Mean ms.	Median ms.	Max. ms.	St.dev.
100	2148	134	141.1	140.2	148	44.4
250	2150	133	140.3	139.6	149	43.8
1000	2146	130	139.9	137.9	153	43.5
10000	2151	128	139.8	137.4	155	43.4

As can be observed from Table 5.11, the best schedule that was obtained by randomly placing activities in the precedence feasible order is the one with the makespan of 128 weeks. On average, the duration of randomly generated schedules in all cases varied from 135 to 145 weeks. The worst schedule that was generated randomly had the makespan of 155 weeks.

Optimised schedules. The results of deterministic scheduling experiments are summarised in Table 5.12. It is worth mentioning that in the given table the optimal solution is the one that has the same makespan as referral solution. In

the context of this experiment, “optimal solution” is the one with a makespan of 113 weeks.

Table 5.12 - Summary of DSCCS performance results for deterministic scheduling

Parameter	Min.	Mean	Median	Max.
Makespan	113	116	115	120
Number of found optimal solutions	0	3	3	5
Computational time (s)	23.9	26.4	25.2	31.3

On average, the proposed case study was relatively easy to solve for the algorithm, as it managed to find the reference solution with a makespan of 113 weeks in roughly of 75% of the total amount of runs. The worst schedule that was obtained by the DSCCS during this experiment had a makespan of 120 weeks. Moreover, in some runs, the algorithm was not able to obtain solution with referral makespan, hence the number “0” in Table 5.12. The average amount of found best solutions has not changed and its values are the same as they were in the previous experiment.

As can be seen in Table 5.12, in some cases the algorithm was not able to obtain solution with referral makespan. Nevertheless, the number of found solutions with the makespan of reference solution (if any were found at all) varied from 1 to 4.

5.1.4.2 Stochastic Scheduling

Randomly-generated schedules. Similarly to what was done in Section 5.1.4.1, performance evaluation of the algorithm on the proposed optimisation model and case study in stochastic scheduling mode begins with the generation of randomly-created feasible schedules. Results of this procedure are summarised in Table 5.13.

Table 5.13 - Durations of randomly-generated stochastic schedules

Size	σ_s*	Min. ms.	Mean ms.	Median ms.	Max. ms.	St. dev.
100	988	119	127.1	124.8	139	37.2
250	1001	117	126.9	124.3	137	39.9
1000	1009	112	126.5	122.9	140	41.8

10000	1015	112	126.5	122.1	146	42.4

Makespan of the randomly-generated schedules in stochastic mode demonstrate that due to the abilities of the resources to influence the durations of activities, given they have enough experience, the average makespan of random schedules in comparison to the deterministic mode has decreased from ~140 to ~127, which is equal to roughly 10%. The shortest makespan of the randomly-generated schedule is 112 weeks, a 12.5% improvement in comparison to standard deterministic schedules. This is due to the fact that in deterministic scheduling experiments learnabilities and efficiencies of resources are not taken into account, hence durations of activities are always constant.

Optimised schedules. For optimisation in stochastic mode, same parameter configurations were used as in deterministic mode. The results of these experiments are summarised in Table 5.14. It is worth mentioning that in the given table the optimal solution is the one that has the same makespan as referral solution. In the context of this experiment, “optimal solution” is the one with a makespan of 97 weeks.

Table 5.14 - Summary of DSCCS performance results for stochastic scheduling

Parameter	Min.	Mean	Median	Max.
Makespan	97	103	101	108
Number of found optimal solutions	0	3	3	5
Computational time (s)	43.4	56.9	51.2	69.1
Resource efficiency	0.0658	0.2224	0.284	0.4219

The best-optimised schedules of the case study project in stochastic mode had the makespan of 97 weeks. In comparison to the best schedules received in deterministic scheduling mode, the makespan of which was 113 weeks, the total duration decreased by 15%.

Similarly to deterministic experiment, in some runs, the algorithm was not able to obtain solution with referral makespan, hence the number “0” in Table 5.14. The average amount of found best solutions has not changed and its values are the same as they were in the previous experiment.

Due to the additional operations that algorithm had to make, such as calculations of new activity durations and resource efficiency balancing, DSCCS

computational time has doubled and in some cases even tripled. On average, it has increased from 26.4s to 56.9s.

The resource efficiency of final solutions in the majority of cases varied in the range of 0.2 to 0.25. The most well-balanced solution had the total resource efficiency of 0.0658.

5.1.4.3 Parameters Influence

Lastly, to conclude this case study, this section analyses how the variance of the DSCCS parameters affects the performance criteria considered in both deterministic and stochastic scheduling modes.

Effect of the population size. Graphs presented in Figure 5.5, Figure 5.6, and Figure 5.7 demonstrate the influence of the population size m on the quality of the obtained solutions, the amount of obtained reference solutions and computational time, respectively, in deterministic and stochastic modes. The value of m in the presented results is set in the range of [10, 250]. The values of other parameters are set to optimal values that were identified by *irace* package from Table 5.10.

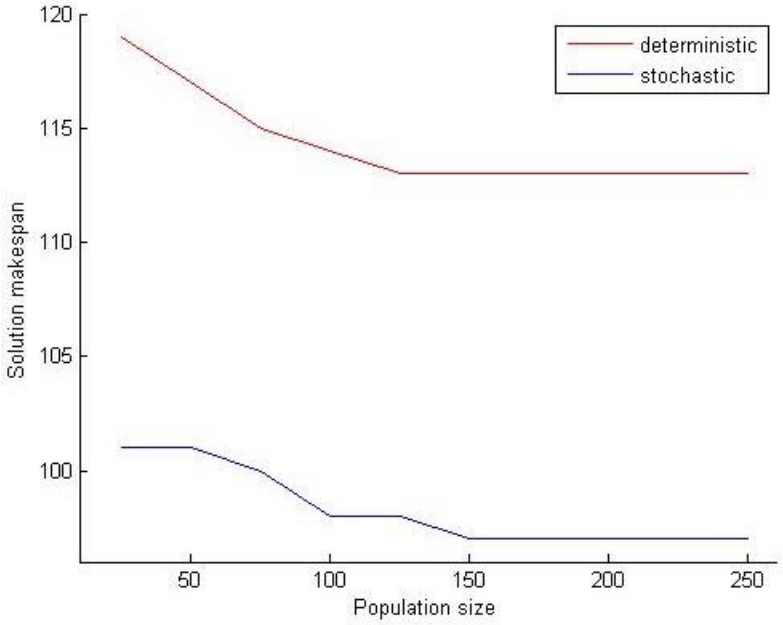

Figure 5.5 – Effect of m on the solution makespan

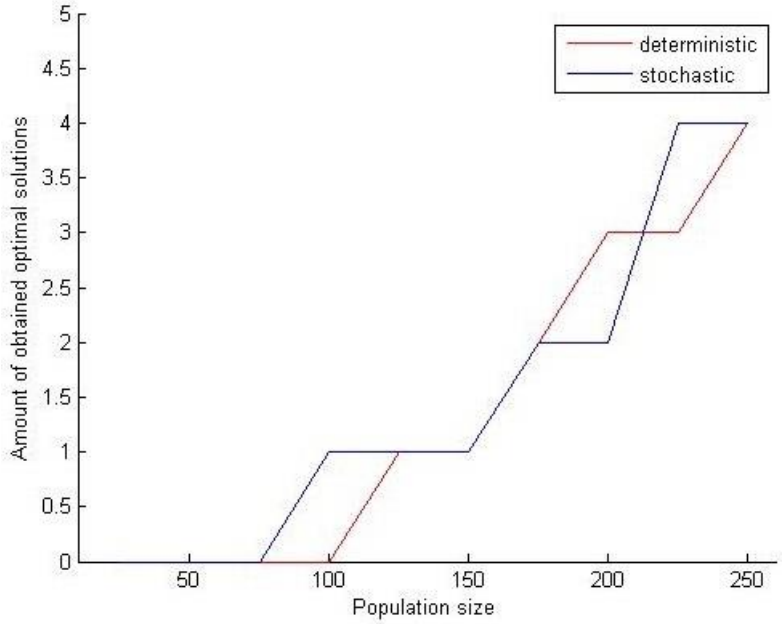

Figure 5.6 – Effect of m on the amount of obtained results

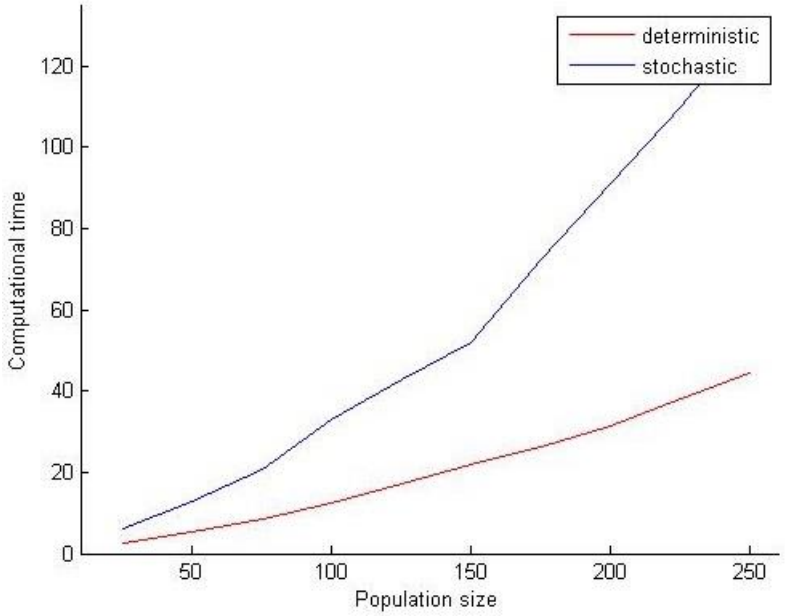

Figure 5.7 – Effect of m on the total computational time

As can be observed from the above-presented graphs, the population size m has a direct impact on all performance criteria. Its increase has a positive impact on the performance and success rate of the algorithm, and negative impact on total computation time, as with higher population the algorithm has to do more operations. The minimal value of m, which the DSCCS was able to obtain with the reference solution (in combination with other parameters), was 125 in deterministic mode and 150 in stochastic mode.

Effect of the species distance. Graphs presented in Figure 5.8, Figure 5.9, and Figure 5.10 show the influence of the species distance σ_s on the quality of

obtained solutions, the amount of obtained optimal solutions and computational time, respectively, in deterministic and stochastic scheduling modes. The values of σ_s in the presented graphs are set in the range of [σ_s*/5, σ_s*]. The stopping criterion was set to 50000 objective evaluations, whereas the values of other parameters coincide with those in Table 5.10.

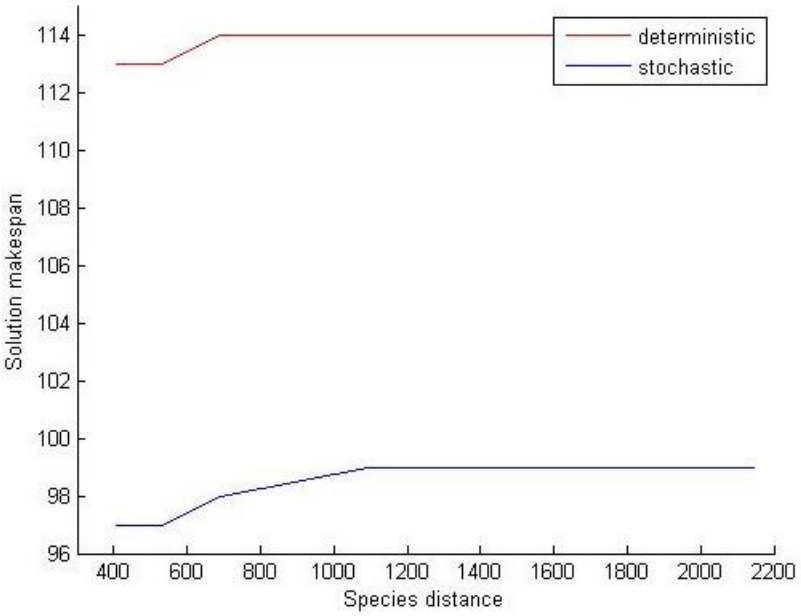

Figure 5.8 – Effect of σ_s on the solution makespan

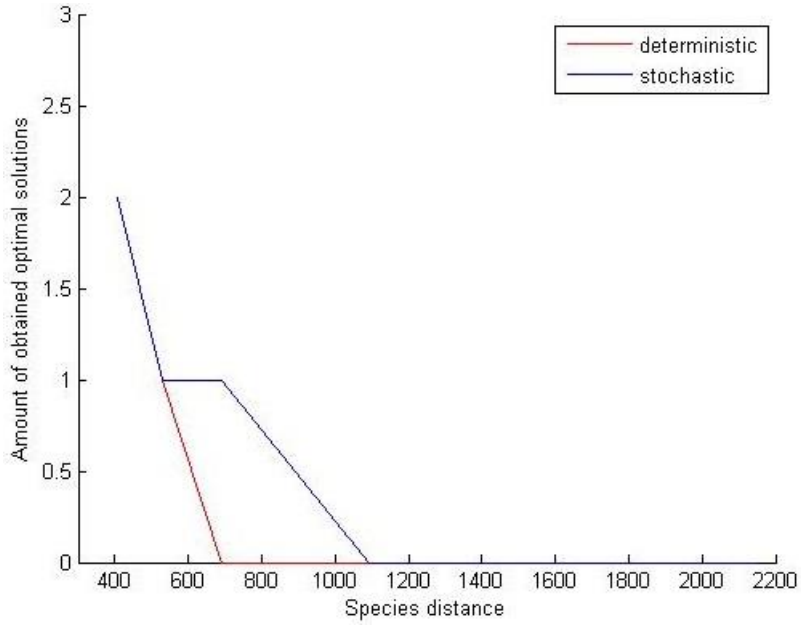

Figure 5.9 – Effect of σ_s on the amount of obtained results

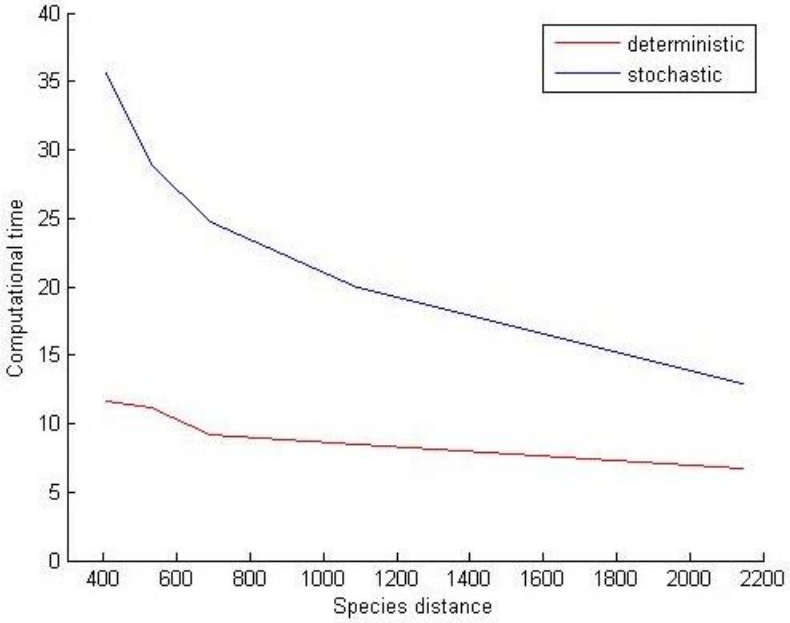

Figure 5.10 – Effect of σ_s on the total computational time

The effect of σ_s in this experimental setup coincides with the results of the DSCCS experimental evaluation conducted in Section **Error! Reference source not found.**. The smaller values of σ_s result in more species formed in the overall population in which leads to greater chances of finding the reference solution. As can be noted from Figure 5.8 and Figure 5.9, with smaller σ_s values, the algorithm is capable of finding more solutions with the best-known makespan. However, all of this also results in higher computational overhead, which is confirmed by the graphs.

Effect of the stopping criterion. Graphs presented in Figure 5.11, Figure 5.12, and Figure 5.13 demonstrate the influence of the stopping criterion on the quality of obtained solutions, the amount of obtained optimal solutions and computational time, respectively, in deterministic and stochastic scheduling modes. The values of stopping criterion in the presented graphs are set in the range of [1000, 10000]. The values of other parameters coincide with those in Table 5.10.

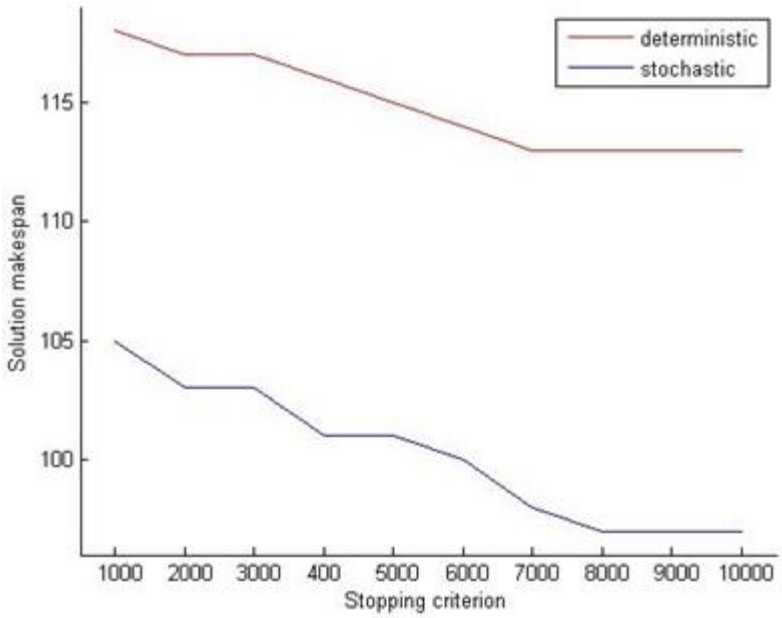

Figure 5.11 – Effect of stopping criterion on the solution makespan

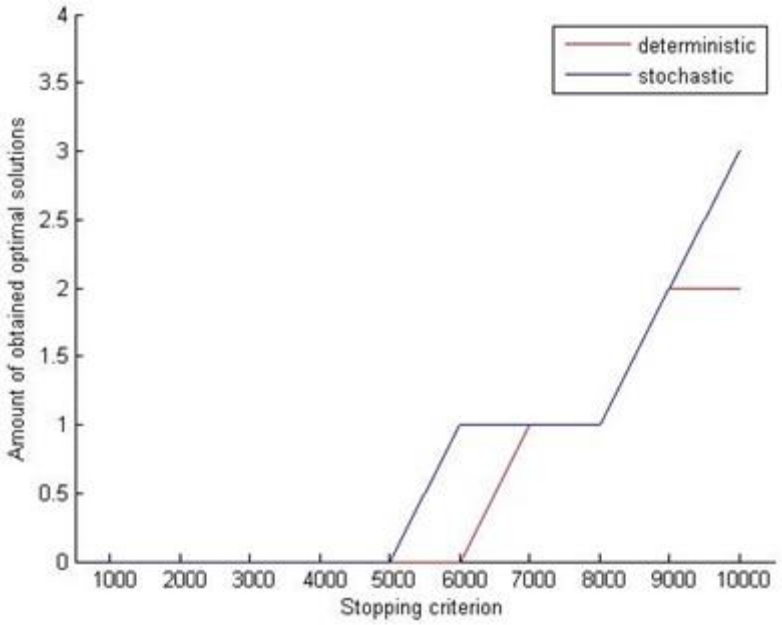

Figure 5.12 – Effect of stopping criterion on the amount of obtained results

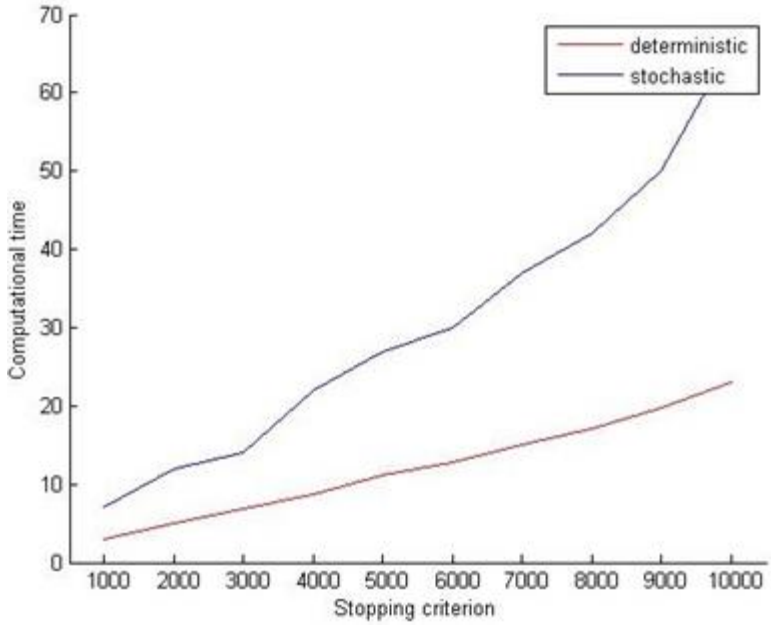

Figure 5.13 – Effect of stopping criterion on the total computational time

From the above-presented results, it is easy to see that the effect of stopping criterion on the performance evaluation criteria is very similar to the one of the population size m. The higher value of stopping criterion results in the better quality of received solutions and higher computational overhead.

5.1.4.4 Comparative Analysis

For the comparative analysis two solutions s_1 and s_2 with reference makespans of 113 and 97 weeks from deterministic and stochastic experimental setups, respectively, are selected for further analysis and investigation of the change of activity durations. A complete description of solutions is provided in Figure 5.14 and Figure 5.15, where each figure depicts execution schedule of the respective solution. Each of the figures is divided into six sections, where each section corresponds to a particular resource type. Resources are aligned in the same order as they appear in Table 5.5 (i.e. section 1 of each of the figures corresponds to R_1). In each of the sections, rows represent available resource units and each block in the row stands for an activity, each colour shade corresponding to a particular activity. Each of the sections shows the consumption of resource units by activities over the time scale. The ELs of the selected solutions are presented in Table 5.15. EL of s_1 consists of 41 events, while EL of s_2 consists of 36 events.

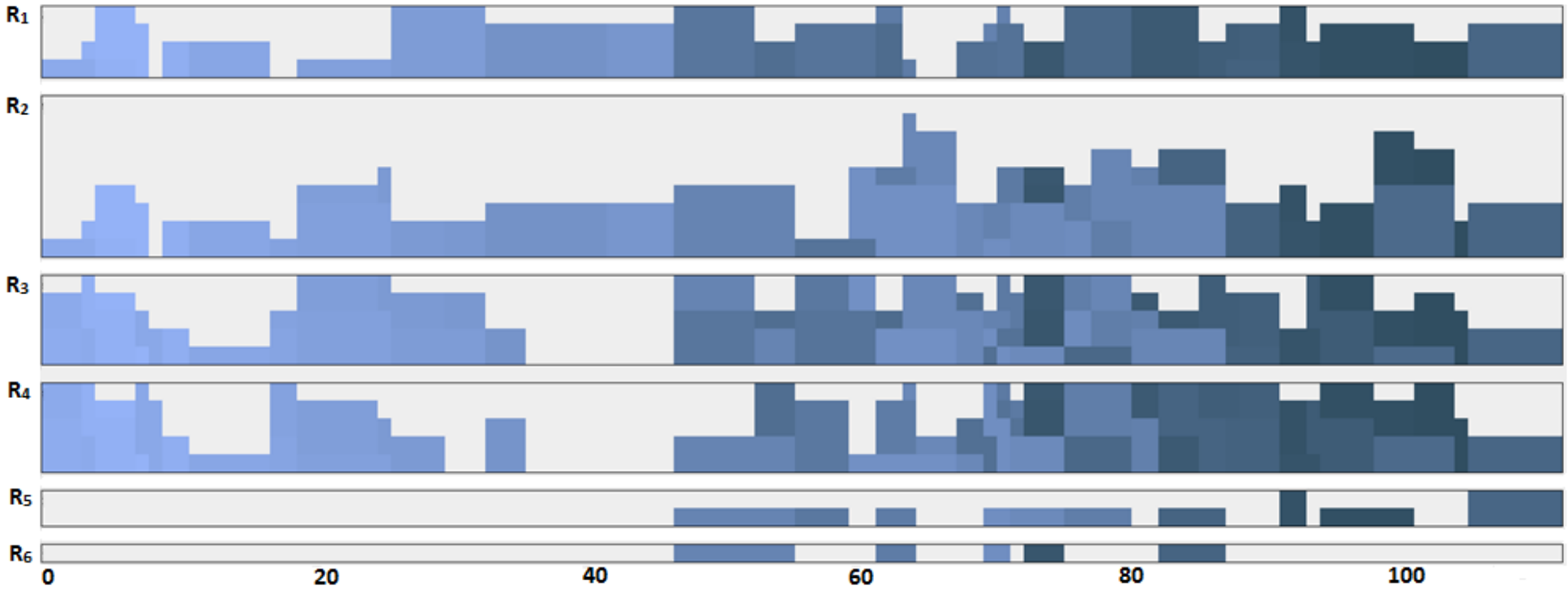

Figure 5.14 – Sample optimal deterministic schedule

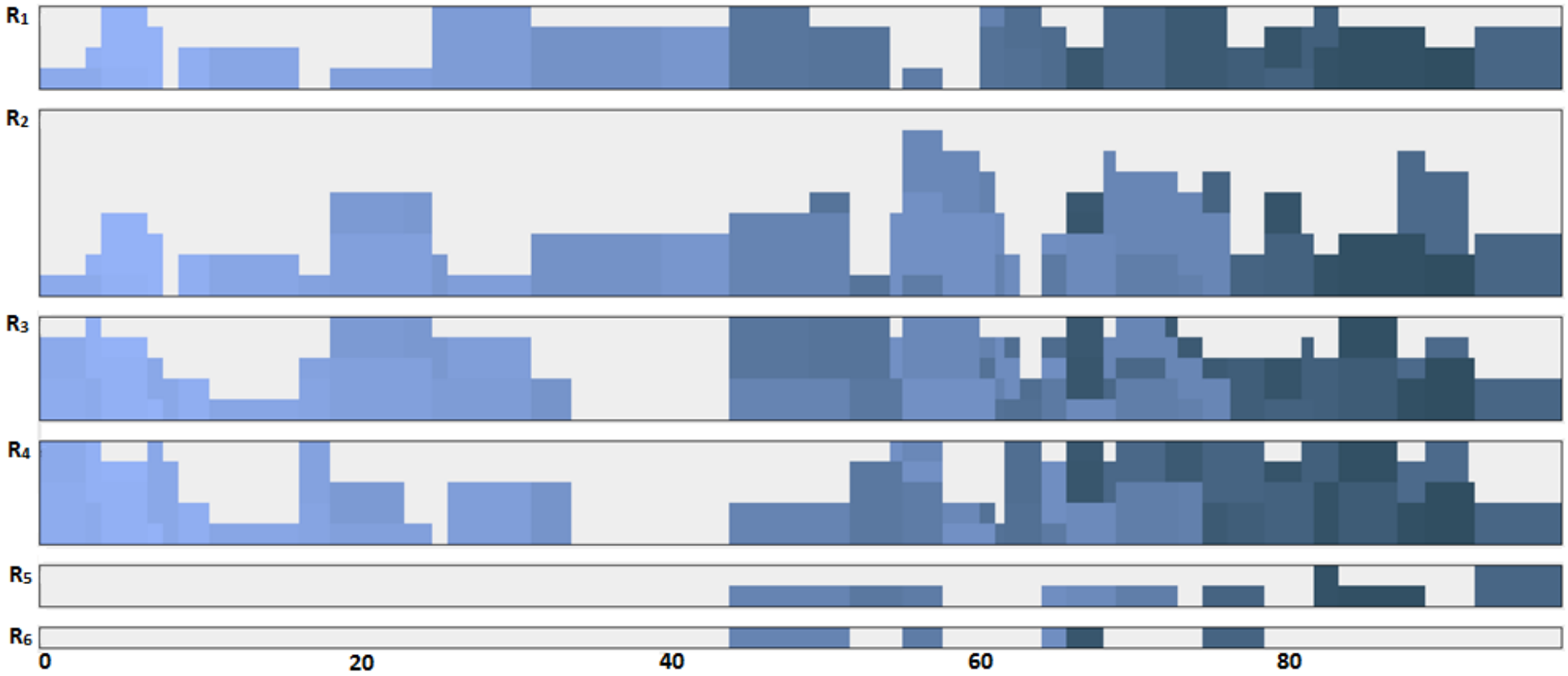

Figure 5.15 – Sample optimal stochastic schedule

Table 5.15 - Event list representations of sample solutions

#	Event list representation
s_1	{[4, 5] [2] [1] [6] [3] [7] [8, 9] [10, 11] [13] [12] [14] [17] [16] [15] [31, 24] [35] [32, 30] [18] [36, 28] [22] [34] [19, 25] [33, 29] [21] [47, 46] [38, 20] [27] [23] [45] [26] [40] [43] [41] [42] [48] [49] [44] [50] [37] [51] [39]}
s_2	{[4, 5] [2] [1] [6] [3] [7] [8, 9] [10, 13] [14] [12] [11] [17, 16] [15] [24, 31] [32] [30] [18] [28, 22] [33, 25] [20] [34, 35] [29, 19, 36] [21, 47, 46] [38, 23] [27] [45] [26] [40] [43] [41, 48] [42] [49] [44, 50] [37] [51] [39]}

Both solutions have optimal makespan for their respective scheduling modes and, as the result, their activity sequences bear many similarities. The first halves of both schedules (i.e. execution of stage 1 and stage 2 activities), due to the linearity of these stages, are almost identical and the overall activity sequencing and resource allocation can be regarded as the same. Moreover, from the presented examples, it is easy to see that the majority of work that was done by the algorithm in terms of scheduling was during the sequencing of stage 3 and stage 4 activities, mainly due to their complex precedence relationships and variety of possible scheduling combinations. Throughout the execution of the project, resources were used sparsely, i.e., not each unit was needed during each time period. Nevertheless, for each of the resource types, there are also periods of very high activity and intense use.

Lastly, Figure 5.16 displays the decrease of activity durations in percentage throughout the execution of the project on the example of s_2.

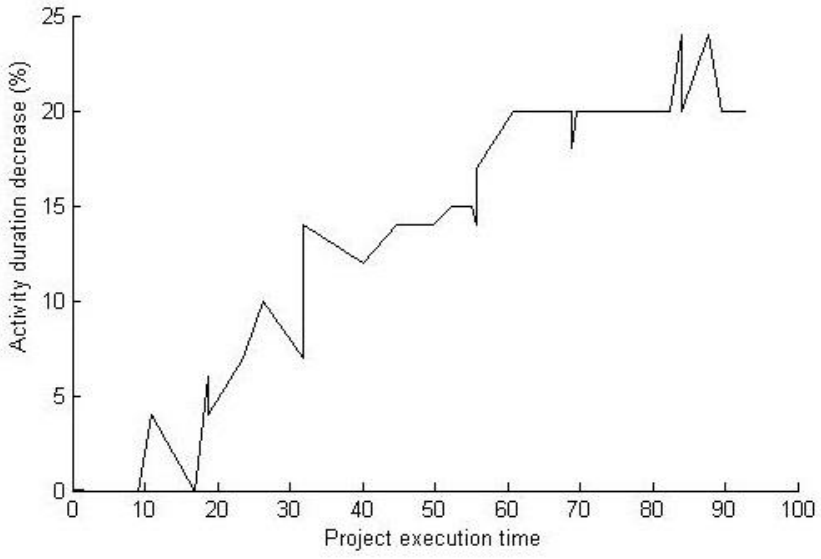

Figure 5.16 – Activity durations decrease throughout the project execution

From the presented results it can be observed that as the execution of project goes on, activities that were started at later time periods benefited more than those that were started earlier. As can be noted from the presented graph, due to the fact that resources have different efficiency and learnability coefficients, and imperfect resource efficiency balancing, some of the activities have benefited more than the others. For example, at time period 11 duration of one of the activities was shortened by 4%, however, the duration of activity that was started at later time period 17 remained the same. Towards the end of project's execution, activity durations were shortened on average by 20%. The biggest decrease in the activity duration was by 24%.

5.2 Further Performance Comparison

In order to furtherly assess performance of the DSCCS on the proposed optimisation model, several among the most popular algorithms in the literature for the deterministic RCPSP have been implemented. Each of the implemented algorithms, along with DSCCS, were used to run edited RCPSP benchmark instances from PSPLIB (Kolisch & Sprecher, 1997). The obtained results are then compared and analysed.

5.2.1 Algorithms

For the selection of the algorithms for this experiment, the following criteria has been considered:

- Type of the applied SGS
- Computational performance
- The simplicity of implementation

Due to the specifics of the optimisation model to which these algorithms are going to be applied, the standard variations of SGS are not applicable for this scenario. Instead, a variation of the serial SGS is going to be used to produce schedules with time- and resource-dependant activity durations. For further description of the applied SGS refer to Section 5.1.2.1. Since the proposed SGS is a derivative of serial SGS, it is desirable that the algorithm selected for the comparison would also be based on the serial SGS.

As the result of the selection process, the following algorithms have been chosen to implement:

- SA, implementation developed by Bouleimen and Lecocq (2003)

- GA, version of Hartmann (1998)
- TS, version of Nonobe and Ibaraki (2002)

In the past, these algorithms were demonstrated to be very effective and competitive in solving the deterministic RCPSP. In the experimental evaluation of multiple heuristics for the RCPSPs, done by Kolisch and Hartmann (2006), the above-mentioned algorithms were applied to solve sets of benchmark instances from PSPLIB. As the result of evaluations, the algorithms ended up being in top 10 among 60 different heuristics. The key characteristics of these algorithms are summarised in Table 5.16. For more detailed information refer to the original works.

Table 5.16 - Brief summary of implemented algorithms

Name	SGS	Solution representation	Genetic operators
SA	Serial with FBI	AL	Shift move operator
GA	Serial	AL	Two-point crossover, activity swap
TS	Serial with FBI	AL	Shift move operator

The application of these algorithms for the proposed optimisation model will only result in the change of SGS. Other key elements will remain the same.

5.2.2 Experiment Setup and Parameter Choices

To evaluate performance of DSCCS on the proposed optimisation model and compare its performance against other algorithms, subsets of the standard benchmark instances from PSPLIB are used. For this experiment, four datasets are created, where each dataset contains benchmark instances from J30 (total of 480 instances), J60 (total of 480 instances), J60 (total of 480 instances), and J120 (total of 600 instances) sets and consists of problem instances with 30, 60, 90, and 120 activities, respectively. Due to the large variety of available problem instances, in order to fully test algorithms on problems with different structural parameters, from each of the PSPLIB problem sets event 10^{th} benchmark instance is selected. Therefore, in total 204 instances are selected: 48 instances from J30 set, 48 instances from J60 set, 48 instances from J90 set and 60 instances J120 set.

Such approach ensures that all combinations of structural parameters are covered and all aspects of the algorithm performance are tested. To model the effect of resource learnability and experience, for each of the resource types, the following is assumed:

- R_1: e_k = 0.15, l_k = 35
- R_2: e_k = 0.20, l_k = 25
- R_3: e_k = 0.20, l_k = 25
- R_4: e_k = 0.15, l_k = 15

The remaining parameters of each of the benchmark instances remain unchanged.

In the experiments, each algorithm was used to run each of the benchmark instances 25 times and the results for each instance will be averaged. For this procedure, the stopping criterion is going to be set to 50000 objective evaluations. Other parameters of the implemented algorithms are going to be set to the same values as were used in the original experiments performed by their respective authors. The DSCCS parameters for this evaluation are going to be set to those that are in Table 4.17.

For the performance evaluation in this experiment two criteria are considered:

- Deviation from CP
- Computational time

The first criteria, deviation from CP, reflects the difference in the duration of the obtained schedule from its CP, which is obtained by scheduling all activities ignoring the resource constraints and is obtained as follows:

$$dev = \frac{result - CP}{CP} * 100\% \tag{36}$$

The second criteria, computational time, reflects the total amount of time that was required for the algorithm to solve the benchmark instance. In traditional evaluations of the RCPSP algorithms, the computational time is never taken into account, primarily because of the different experimental setups and algorithm implementations. However, since in this experiment all algorithms are going to be run on the same machine and will follow the same procedure of performance evaluation, their computational time can be adequately measured.

Due to the inability of the implemented algorithms to obtain multiple solutions, the resource efficiency balancing objective in this procedure is not going to be optimised.

5.2.3 Comparative Analysis

The experimental results of the algorithm's performance evaluation are summarised in Table 5.17, Table 5.18, Table 5.19, and Table 5.20 for J30, J60, J90, and J120 sets, respectively. The first column denotes the name of the algorithm, column "Author(s)" shows the name of the original author. Column "Dev. (%)" shows average deviation of solutions from the critical path (CP). Column "Comp. time" displays the average computational time required to solve each of the benchmark instances. All results in the tables are sorted with respect to the average deviation. In accordance to standard RCPSP performance evaluation experiments, here as the main performance factor is only considered deviation from optimal solutions, whereas computational time is left out and provided strictly as a reference.

Table 5.17 - Experimental evaluation results for J30 dataset

Algorithm	Author(s)	Dev. (%)	Comp. time
DSCCS	Bibiks et al.	0.00	29.7
TS	Nonobe and Ibaraki (2002)	0.06	21.3
SA	Bouleimen and Lecocq (2003)	0.08	17.5
GA	Hartmann (1998)	0.09	23.6

Table 5.18 - Experimental evaluation results for J60 dataset

Algorithm	Author(s)	Dev. (%)	Comp. time
DSCCS	Bibiks et al.	4.36	63.1
SA	Bouleimen and Lecocq (2003)	6.81	38.6
TS	Nonobe and Ibaraki (2002)	7.25	41.2
GA	Hartmann (1998)	7.49	45.3

Table 5.19 - Experimental evaluation results for J90 dataset

Algorithm	Author(s)	Dev. (%)	Comp. time
DSCCS	Bibiks et al.	13.93	95.5
GA	Hartmann (1998)	15.85	74.1
TS	Nonobe and Ibaraki (2002)	16.01	71.9
SA	Bouleimen and Lecocq (2003)	16.13	64.4

Table 5.20 - Experimental evaluation results for J120 dataset

Algorithm	Author(s)	Dev. (%)	Comp. time
DSCCS	Bibiks et al.	25.14	130.9
GA	Hartmann (1998)	29.18	113.3
TS	Nonobe and Ibaraki (2002)	30.01	112.9
SA	Bouleimen and Lecocq (2003)	32.38	101.2

The above-presented results show that by managing to obtain lowest deviation from optimal solution in all experiments, DSCCS achieves the highest performance between all compared algorithms for all datasets. The performances of other implemented algorithms are in line with the performance evaluations that were by done by Kolisch and Hartmann [22]. These results demonstrate that the application of the algorithm to the proposed model does not impact its performance. It also was worth mentioning that in all experiments DSCCS was able to obtain from three to six solution candidates, whereas other algorithms, due to their limitations, could obtain only one solution.

Computational time, however, shows a different picture. Here, DSCCS demonstrated the worst result, mainly due to the additional computational overhead that is caused by the species conservation procedure. Among all tested algorithms, the fastest to solve all benchmark instances was SA. The main reason for such fast computational speed is the work only with one solution and reduced amount of operations that it makes at each iteration. The computational time of TS is close to the one of the SA, primarily because of the fact that both these methods are trajectory-based. Computational time of GA is somewhere in the middle between the ones of SA and DSCCS.

5.3 Summary

The work in this chapter focused on the application of the DSCCS on the optimisation model proposed in Chapter 3 and activity scheduling with varying resource efficiencies. For this, two sets of experiments are carried out: scheduling and sequencing of the activities of the real practical project, and execution of the edited PSPLIB benchmark instances and subsequent comparison of the received results with the results obtained by other implemented algorithms.

For the first experiment, the HARNet project is selected as the practical example, which is then used to demonstrate the applicability of the proposed

optimisation model for scheduling development projects, and analyse the behaviour of the DSCCS when applied to the real-world project. HARNet represents a large-scale aeronautical project which relied on the collaboration of many partners and consisted of many subprojects (WPs). In this case study, the DSCCS is applied to schedule the activities of the WP9, mainly due to the reasons that people who have worked on this sub-project had very little of relevant experience and as the project went on, their effectiveness improved. The WP9 consisted of 51 activities and 6 resource types. The resource from 1 to 4 represent a group of researchers, whereas resources 5 and 6 represent specialised equipment. Because of that, only resources 1-4 can impact the activity durations. To analyse the difference between deterministic and stochastic scheduling, and study the resource experience gain and its effect on the activity durations, the DSCCS is applied to solve two instances of the case study: deterministic, in which activity durations do not vary, and stochastic, in which the activity durations depends on the execution time and applied resources. The reference schedule received in deterministic mode had the makespan of 113 weeks. The reference makespan of stochastic schedule was 97 weeks. Further, two best schedules (one from the deterministic mode and one from stochastic) are selected for comparative analysis. The analysis has shown close to the end of the project's execution, the duration of the activities has reduced on average by 20%.

For the second experiment, three among the most popular methodologies for the deterministic RCPSP are implemented and are applied to solve the PSPLIB benchmark instances. To correlate the benchmark instances to the proposed optimisation mode, the instances are modified to include additional parameters for resource efficiency and learnability. In total 204 instances are selected and are divided into four datasets. Implemented algorithms, along with the DSCCS, are applied to solve each of the benchmark instances. To evaluate the performances of the algorithms, two criteria are considered: average deviation from CP, and average computational time. As the result of the experimental evaluation, the DSCCS showed the best level of performance among all algorithms, however, at the same time, it had the worst computational time, mainly due to the additional computational overhead that was caused by the application of the species conservation technique.

Chapter 6 Conclusions and Future Work

This chapter of the thesis concludes the work that was done during this PhD study, summarises the main achievements and accomplishments, and identifies possible areas for further improvement.

6.1 Conclusions

This PhD work builds on Cuckoo Search (CS) and Flower Pollination Algorithms (FPA) algorithms and extends them to address challenges in optimising large-scale project schedule in an uncertain environment, subject to multiple constraints such as limited resource capacities and strict precedence relationships between activities. Several novel concepts have been proposed, implemented and tested during this PhD study.

a) Derivation of Discrete Cuckoo Search (DCS) and Discrete Flower Pollination Algorithm (DFPA), respectively, to solve RCPSP in discrete domain

DCS and DFPA were derived by adapting CS and FPA to solve discrete RCPSP rather than RCPSP in the continuous domain, which CS and FPA were originally developed, through reinterpretation of their key elements: solution representation scheme, solution improvement operators and Lévy flight. For the solution representation scheme activity list was chosen, as it is the most common representation scheme for the RCPSP and a large variety of operators has been developed. In the review on state-of-the-art heuristics for the RCPSP by Kolisch and Hartmann (2006) 23 out of 27 algorithms operated on activity list solution representation scheme. For solution improvement operators, pairwise exchange and shift operators were selected, as combination of these operators seemed to work best when integrated into Lévy flight: for smaller Lévy number (i.e. small step) a number of shift operations was performed on a solution, whereas for big numbers (i.e. large step) pairwise exchange was executed.

Both algorithms were evaluated in accordance to the standardised tests defined by Hartmann et al. (2000) by running all benchmark instances of J30, J60 and J120 sets from PSPLIB for 1000, 5000 and 50000 objective evaluations. Results showed that both algorithms were capable of solving discrete problems and both showed relatively good performance and managed to outperform some non-hybrid metaheuristics against which they were compared, such as Tabu

Search (TS) and Genetic Algorithm (GA). In particular, DCS appeared in top 1 for J30, J60, and J120 sets, whereas DFPA was outperformed by TS in J30 tests by 0.01% and by GA in J120 by 0.25%. However, both algorithms suffered from several drawbacks that affected their performance: reliance on inefficient solution representation scheme (activity list) and utilisation of random-based solution improvement operators (pairwise exchange and shift operators). This leads to the development of the IDCS algorithm.

b) Improved Discrete Cuckoo Search (IDSC) algorithm

The main disadvantage of the activity list solution representation scheme in DCS and DFPA is the representation of a single schedule by multiple representation schemes. Because of that, if two activities have identical starting times, then interchanging their positions will not bring any changes to a solution, which, in turn, results in wasted operation. To address this issue, IDCS replaced activity list by event list. The main distinctive feature of event list from other solution representation scheme is that activities with identical starting times are grouped into events. In some circumstances, such events can comprise activities sharing common project characteristics, such as having the same predecessors and/or successors. Moreover, the introduction of the new solution representation scheme resulted in smarter and less random operators to be used to their advantages: event move exploits shared network characteristics of events for more efficient perturbations, whereas event crossover was used to combine useful problem-specific information extracted from the parents for generating high-quality offspring. The performance of IDCS was again evaluated on sets of benchmark instances from PSPLIB and compared against other algorithms for the RCPSP. The results showed that IDCS greatly improved the performance of DCS and DFPA. In addition, IDCS outperformed all other metaheuristics for J120 sets and its performance was among the top five for J30 and J60 sets.

During the validation of IDCS on single benchmark instances, it was found that IDCS is able to obtain multiple solutions for one problem instance. In fact other researchers, for example Czogalla and Fink (2009), also pointed out that RCPSPs exhibit complex multimodal fitness landscapes; hence, for one instance of the problem, several optimal solutions might exist. Multiple solutions are beneficial in that they can eliminate premature convergence to local optima and can sometimes lead to more innovative outcomes, such as more efficient or well-balanced schedules. Nevertheless, at the time of writing, this property had only

been addressed in one publication (Pérez, Posada, & Lorenzana, 2015). As a result, research effort thereafter was devoted to explore multimodal optimisation techniques such as the Species Conservation (SC) technique to combine with IDCS to obtain multimodal solutions for RCPSPs.

c) Development of Discrete Species Conserving Cuckoo Search (DSCCS)

In order to explore the potential of IDCS to obtain multimodal solutions, techniques for multimodal optimisation were reviewed. Among the different techniques, SC was chosen for its simplicity and effectiveness as it only relies on distance metric (Euclidean distance) to estimate the similarity between solutions. However, due to the specifics of the RCPSP and its discrete search space, SC could not be directly applied. Therefore, in order to be adapted, the Euclidian distance metric needed to be replaced with a similarity metric specific for the RCPSP. Czogalla and Fink (2009) in their analysis of the RCPSP fitness landscape reviewed various distance measure techniques. The authors conducted a series of experiments on the sets of benchmark instances from PSPLIB. As the conclusion of their analysis, the authors noted that algorithms that operated on the deviation distance measure (Reeves, 1999) tend to produce better results. Moreover, Chen et al. (2010) and Paraskevopoulos et al. (2012) used the abovementioned distance measure in their algorithms, which confirms its applicability and effectiveness. As the conclusion of these analyses, deviation distance measure was chosen as the main method of similarity measure in adaptation of SC to the RCPSP.

SC technique was then integrated into IDCS to form DSCCS. In DSCCS the whole population was divided into smaller subpopulations (i.e. species). Each subpopulation had devoted region of search space and was centred on the fittest solution (i.e. dominating individual). Having multiple sub-population ensures that diversity of population is kept high at all stages and significantly reduces chances of falling into local optima trap.

Similarly to other developed algorithms, the performance of DSCCS is assessed on the sets of benchmark instances for RCPSP. In comparison to IDCS, the performance was slightly degraded for J120 and the algorithm moved from top 1 to top 5. For J30 and J60 sets its performance stayed on the same level. The lower performance can be explained by the fact that due to having a need to maintain several sub-populations in parallel and being restricted by the amount of objective evaluations that can be performed, the amount of times Lévy

flight can be applied to improve one solution has decreased. Nevertheless, such trade-off in performance is understandable and has been discussed in many publications (Wolpert & Macready, 1997). However, DSCCS achieved what other algorithms could not achieve in finding multiple solutions for RCPSPs.

During algorithm testing, depending on the test instance, the number of candidate solutions varied from 1 to 22, 1 to 60, and 1 to 21 for J30, J60, and J120 test instances, respectively. The number of solutions primarily depended on the value of the species distances σ_s: for smaller value of σ_s the algorithm was capable to obtain more solutions and vice versa. It is also worth mentioning that the DSCCS was able to find multiple optimal (or best-known) solutions in the majority of all test instances with high success rate.

d) Extension of the RCPSP by introduction of efficiency and learnability factors to resources

In order to take advantage of the RCPSP multimodal properties, the HARNet Project Management Problem (HPMP) is proposed. HPMP can be characterised as a special case of the RCPSP and it represents a model for scheduling large and complex projects that are new to their execution environment. The main difference of HPMP from other variants of the RCPSPs is that throughout the project execution, as the resources are consumed by activities, the resources gain experience which then can influence the durations of activities. The rate at which resources acquire experience is defined by their learnability coefficient. The maximum amount of time by which the duration of the activity can be reduced is defined by the effectiveness coefficient. To take the advantage of RCPSP's multimodal property, for successful completion HPMP considers optimisation of two objectives: primary – makespan minimisation, and secondary – resource efficiency balancing. The optimisation of these objectives is achieved as follows: firstly, the applied algorithm acquires a set of candidate solutions with the shortest makespan; secondly, out of the acquired set, the algorithm chooses the most suitable solution with the lowest resource efficiency-balancing coefficient. Nevertheless, HPMP can also be solved by the methodology developed for standard RCPSP; however, in this case, only one objective can be optimised at a time.

A application scenario HPMP was devoted to the planning and management of the large-scale projects (i.e. software development) where activities were executed through different resources (i.e. groups of researchers and developers)

that have no, or have very little of, relevant experience. In the beginning of the execution of such projects, there is a high level of uncertainties caused by the lack of knowledge and expertise. However, as the project goes on, people practice relevant skills and, as the result, the level of uncertainties vanishes and the duration of activities reduces.

To assess applicability of the proposed optimisation model in real-life scenario and further evaluate performance of the DSCCS two case studies are created. First case study represents a scheduling of the practical project. The considered project is the real-world project that was undertaken a year ago. It consisted of 51 interrelated activities execution of which required 6 types of resources. Four of these resources are groups of people, other two types of resources are specialised equipment. For this case study, two sets of experiments are conducted, which cover the production of normal deterministic and variable stochastic schedules. The results of these experiments are then compared and analysed. For the second case study, several most popular methodologies for the RCPSP are implemented. Further, the performances of the implemented methodologies, along with the DSCCS are evaluated on sets of the edited benchmark instances. The main difference of these benchmarks from their standard variants is the inclusion of two additional parameters needed for the proposed optimisation model: learnability and effectiveness coefficients. As the results of evaluation, the DSCCS has greatly outperformed all other implemented algorithms, however, its computational time the worst among all. Nevertheless, these case studies demonstrated the operability and effectiveness of the proposed optimisation model: showed how the activity durations may decrease depending on the time of execution and applied resource, and verified the relative ease of application RCPSP methodologies to it. Moreover, they confirmed the great performance of the DSCCS, its capability to obtain multiple solutions, and its applicability in scheduling of real-life projects.

6.2 Future Work

Even though the case studies conducted in Chapter 5 demonstrated the effectiveness of the proposed optimisation model and great performance of the DSCCS, there are still areas for the possible improvements.

6.2.1 Performance Improvement

By managing to outperform the majority of state-of-the-art heuristics in benchmark tests, IDCS and DSCCS showed competitive level of performance. Nevertheless, despite this, there still were several methodologies with better performance results, which shows that there is still some room for improvement left.

6.2.2 Optimisation of Additional Objectives

Traditional RCPSPs consider optimisation of only one objective: makespan minimisation. The optimisation model proposed in this thesis introduces optimisation of the second objective: resource efficiency balancing. However, in the real-world project, sometimes optimisation of these two objectives is not as vital as the minimisation of the overall cost of the project. In order to introduce the objective of cost minimisation to the problem, several conditions need to be introduced:

- With the execution of each of the activities needs to be associated new parameter that will specify the cost of execution of this activity at certain period of time;
- The durations of activities need to follow new probability distribution which can randomly reduce or increase their durations;
- Generation of a baseline deterministic schedule, which will be used as the reference for establishing the planned costs of the project.

6.2.3 DSCCS Adaptability to the Problem-Specific Setting

A possible area of improvement for the DSCCS include automatic identification of the key parameters. As the computational experiments have shown, the species distance and population size parameters have very significant impact on the performance of the algorithm and the amount of obtained global optima. The solution search space varies from problem to problem; hence, the optimal values for key parameters will vary as well. To provide the optimal performance, the algorithm has to be able to adapt to the problem at hand and automatically estimate the optimal values for the parameters.

6.2.4 Application of the DSCCS to Other Combinatorial Problems

When applied to solve the RCPSP, DSCCS showed competitive level of performance results as it managed to outperform the majority of the compared to

state-of-the-art methodologies and obtain multiple global solutions for each of the problem instances. However, RCPSP is just one of the combinatorial optimisation problems with multimodal fitness landscape. Examples of other problems include JSSP and TSP, among all. The application of the DSCCS to these problems will primarily consist of the reinterpretation of its key elements: solution representation scheme, genetic operators, and objective function.

References

Aarts, E. H., & Lenstra, J. K. (1997). *Local Search in Combinatorial Optimization.* Chichester, UK: Wiley.

Abbass, H. A., Bender, A., Dam, H. H., Baker, S., Whitacre, J. M., & Sarker, R. A. (2008). Computational scenario-based capability planning. *Proceedings of the 10th annual conference on Genetic and evolutionary computation* (pp. 1437-1444). Atlanta, GA, USA: ACM.

Abdelmaguid, T. F. (2015). A neighborhood search function for flexible job shop scheduling with separable sequence-dependent setup times. *Applied Mathematics and Computation, 260*(1), 188–203.

Agarwal, A., Colak, S., & Erenguc, S. (2011). A neurogenetic approach for the resource-constrained project scheduling problem. *Computers and Operations Research, 38*, 44-50.

Alami, J., Benameur, L., & Imrani, A. A. (2009). A fuzzy clustering based PSO for multimodal optimization. *International Journal of Computational Intelligence Research, 167*, 96–107.

Alami, J., Imrani, A. A., & Bouroumi, A. (2007). multipopulation cultural algorithm using fuzzy clustering. *Applied Soft Computing, 7*(2), 506–519.

Alcaraz, J., & Maroto, C. (2001). A Robust Genetic Algorithm for Resource Allocation in Project Scheduling. *Annals of Operations Research, 102*(1), 83-109.

Alvarez-Valdes, R., Crespo, E., Tamarit, J., & Villa, F. (2008). GRASP and path relinking for project scheduling under partially renewable resources. *European Journal of Operational Research, 189*(3), 1153–1170.

Ando, S., Sakuma, J., & Kobayashi, S. (2005). Adaptive isolation model using data clustering for multimodal function optimisation. *Proceedings of the 7th annual conference on Genetic and evolutionary computation* (pp. 1417-1424). Washington, DC: ACM.

Ando, S., Suzuki, E., & Kobayashi, S. (2005). Sample based crowding method for multimodal optimization in continuous domain. *The 2005 IEEE*

Congress on Evolutionary Computation. Edinburgh, Scotland, UK: IEEE.

Angus, D. (2009). Niching for ant colony optimisation. In *Biologically-Inspired Optimization Methods* (pp. 165-188). Berlin: Springer.

Artigues, C., Michelon, P., & Reusser, S. (2003). Insertion techniques for static and dynamic resource-constrained project scheduling. *European Journal of Operational Research, 149*, 249-267.

Ashtiani, B., Leus, R., & Aryanezhad, M.-B. (2011). New competitive results for the stochastic resource-constrained project scheduling problem: Exploring the benefits of pre-processing. *Journal of Scheduling, 14*(2), 157-171.

Back, T., Fogel, D. B., & Michalewicz, Z. (1997). *Handbook of Evolutionary Computation.* Bristol, UK: IOP Publishing Ltd.

Ballestin, F. (2007). When it is worthwhile to work with the stochastic RCPSP. *Journal of Scheduling, 10*, 153-166.

Ballestin, F., & Leus, R. (2009). Resource-constrained project scheduling for timely project completion with stochastic activity durations. *Production and Operations Management, 18*(4), 459-474.

Bar-Yam, Y. (1997). *Dynamics of Complex Systems.* Cambridge, MA, USA: Addison-Wesley.

Battitti, R., & Protasi, M. (1997). Reactive Search, A history-base heuristic for MAX-SAT. *Journal of Experts Algorithms, 2*, 2-28.

Bean, J. C., Birge, J. R., Mittenthal, J., & Noon, C. E. (1991). Match-up scheduling with multiple resources, release dates and disruptions. *Operations Research, 39*(3), 470–483.

Beasley, D., Bull, D., & Martin, R. (1993). A sequential niche technique for multimodal function optimization. *Evolutionary Computation, 1*(2), 101-125.

Berthaut, F., Pellerin, R., Hajji, A., & Perrier, N. (2014). *A Path Relinking-Based Scatter Search for the Resource-Constrained Project Scheduling Problem.* Octobre: Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation.

Bertsekas, D. (2007). *Dynamic programming and optimal control.* Athena Scientific.

Besten, M. L., Stutzle, T., & Dorigo, M. (2001). Design of iterated local search algorithms: An example application to the single machine total weighted tardiness problem. In *Lecture Notes in Computer Science* (pp. 441–452). Berlin: Springer.

Binato, S., Hery, W. J., Loewenstern, D., & Resende, M. G. (2000). A greedy randomized adaptive search procedure for job shop scheduling. In C. C. Ribeiro, & P. Hansen, *Essays and Surveys on Metaheuristics* (pp. 59-79). Kluwer Academic Publishers.

Birattari, M., Yuan, Z., Balaprakash, P., & Stützle, T. (2010). F-race and iterated F-race: An overview. In M. C. T. Bartz-Beielstein, *Experimental Methods for the Analysis of Optimization Algorithms* (pp. 311–336). Berlin, Germany: Springer,.

Blazewicz, J., Lenstra, J., & Kan, A. (1983). Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics, 5*(1), 11–24.

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys, 35*(3), 268-308 .

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys, 35*(3), 268-308.

Bluma, C., Puchingerb, J., Raidlc, G. R., & Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing, 11*, 4135–4151.

Boctor, F. F. (1996). Resource-constrained project scheduling by simulated annealing. *Internation Journal of Production Research, 34*(8), 2335-2351.

Boese, K. D. (1995). *Cost Versus Distance in the Traveling Salesman Problem.* UCLA CS Department: Technical Report TR-950018.

Bouleimen, K., & Lecocq, H. (2003). A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research, 149*(1), 268–281.

Brucker, P., Knust, S., Schoo, A., & Thiele, O. (1998). A branch and bound algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research, 107*(2), 272–288.

Bui, L. T., Michalewicz, Z., Parkinson, E., & Abello, M. B. (2012). Adaptation in dynamic environments: A case study in mission planning. *IEEE Transactions on Evolutionary Computation, 16*, 190-209.

Calhoun, K., Deckro, R., Moore, J., Chrissis, J., & Van Hove, J. (2002). Planning and re-planning in project and production planning. *Omega, 30*, 155-170.

Campos, V., Glover, F., Laguna, M., & Marti, R. (2001). An experimental evaluation of a scatter search for the linear ordering problem. *Journal of Global Optimisation, 21*, 397-414.

Caro, D. D., & Dorigo, M. (1998). AntNet: Distributed stigmergetic control for communication networks. *Journal of Artificial Intelligence Research, 9*, 317-365.

Cavicchio, D. (1970). *Adapting search using simulated evolution.* Ann Arbor, MI: Ph.D. Dissertation, Univ. Michigan.

Chardaire, P., Lutton, J. L., & Sutter, A. (1995). Thermostatistical persistency: A powerful improving concept for simulated annealing algorithms. *European Journal of Operational Research, 86*, 565–579.

Chen, R. M., Wub, C. L., & Lo, S. T. (2010). Using novel particle swarm optimization scheme to solve resource-constrained scheduling problem in PSPLIB. *Expert Systems with Applications, 37*, 1899-1910.

Chen, W., Shi, Y.-j., Teng, H.-f., & Lan, X.-p. (2010). An efficient hybrid algorithm for resource-constrained project scheduling. *Information Sciences, 180*(6), 1031–1039.

Chetty, S., & Adewumi, A. O. (2013). Comparison Study of Swarm Intelligence Techniques for the Annual Crop Planning Problem. *IEEE Transactions on Evolutionary Computation, 18*(2), 258 - 268.

Cho, S. H., & Eppinger, S. D. (2005). A simulation-based process model for managing complex design projects. *IEEE Transactions on Engineering Management, 52*, 316-328.

Choi, J., Realff, M. J., & Lee, J. H. (2004). Dynamic programming in a heuristically confined state space: A stochastic resource-constrained

project scheduling application. *Computers and Chemicals Engineering, 28*, 1039–1058.

Cioppa, A. D., Stefano, C. D., & Marcelli, A. (2007). Where Are the Niches? Dynamic Fitness Sharing. *IEEE Transactions on Evolutionary Computation, 11*(4), 453 - 465.

Coelho, J., & Tavares, L. (2003). *Comparative analysis of metaheuricstics for the resource-constrained project scheduling problem.* Instituto Superior Tecnico, Portugal: Technical report, Department of Civil Engineering.

Connolly, T. D. (1990). An improved annealing scheme for the QAP. *European Journal of Operational Research, 46*, 93-100.

Czogalla, J., & Fink, A. (2009). Fitness Landscape Analysis for the Resource Constrained Project Scheduling Problem. In *Learning and Intelligent Optimization* (pp. 104-118). Berlin: Springer Berlin Heidelberg.

Czyzak, P., & Jaskievicz, A. (1996). Metaheuristic technique for solving multiobjective investment planning problem. *Controls and Cybernetics, 25*, 177–187.

Davenport, A. J., & Beck, J. C. (2001). *A survey of techniques for scheduling with uncertainty.* Unpublished Manuscript, University of Toronto. Retrieved July 2017, from http://tidel.mie.utoronto.ca/publications.php

Davenport, A., Gefflot, C., & Beck, J. (2001). Slack-based techniques for robust schedules. *Seventh International Conference on Principles and Practice of Constraint Programming*, (pp. 91-105). Paphos, Cyprus.

Deb, K., & Goldberg, D. E. (1989). An investigation of niche and species formation in genetic function optimization. *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 42-50). San Mateo, CA: Morgan Kaufmann Publishers Inc.

Debels, D., & Vanhoucke, M. (2005). A Decomposition-Based Genetic Algorithm for the Resource-Constrained Project-Scheduling Problem. *Operations Research, 55*(3), 457 - 469.

Debels, D., De Reyck, B., Leus, R., & Vanhoucke, M. (2006). A hybrid scatter search/electromagnetism meta-heuristic for project scheduling. *European Journal of Operational Research, 169*(2), 638 - 653.

Della Croce, F. (1995). Generalized pairwise interchanges and machine scheduling. *European Journal of Operational Research, 83*, 310-319.

Dell'Amico, M., Lodi, A., & Maffioli, F. (1999). Solution of the cumulative assignment problem with a well-structured tabu search method. *Journal of Heuristics, 5*, 123–143.

Demeulemeester, E., & Herroelen, W. (2002). *Project Scheduling - A Research Handbook.* Boston: Kluwer Academic Publishers.

Demeulemeester, E., & Herroelen, W. (2011). Robust Project Scheduling. *Foundations and Trends® in Technology, Information and Operations Management, 3*(3-4), 201-376.

Dick, G. (2010). Automatic identification of the niche radius using spatially-structured clearing methods. *IEEE Congress on Evolutionary Computation, 18-23 July 2010.* Barcelona, Spain: IEEE. doi:10.1109/CEC.2010.5586085

Dodin, B. (1984). Determining the K most critical paths in PERT networks. *Operations Research, 32*(4), 859-877.

Dodin, B. (2006). A practical and accurate alternative to PERT. *Perspective in modern project scheduling, 46*, 3-24.

Dong, H., He, J., Huang, H.-K., & Hou, W. (2005). A Mixed Mutation Strategy Evolutionary Programming Combined with Species Conservation Technique. *Lecture Notes in Computer Science, 3789*, 593-602.

Dorigo, M., & Stutzle, T. (2003). The ant colony optimisation metaheuristics: Algorithms, applications, and advances. In F. W. Glover, & G. A. Kochenberger, *Handbook of Metaheuristics* (pp. 251-285). Norwell, MA: Springer US.

Dorigo, M., Caro, G. D., & Gambardella, L. M. (1999). Ant algorithms for discrete optimization. *Artistic Life, 5*(2), 137–172.

Dorn, J., Kerr, R., & Thalhammer, G. (1995). Reactive scheduling: improving robustness of schedules and restricting the effects of shop floor disturbances by fuzzy reasoning. *International Journal of Human–Computer Studies, 42*, 687–704.

Dorndorf, U. (2002). *Project Scheduling with Time Windows – From Theory to Applications.* Berlin: Physica-Verlag.

Dréo, J., Pétrowski, A., Siarry, P., & Taillard, E. (2006). *Metaheuristics for Hard Optimization.* Berlin: Springer-Verlag Berlin Heidelberg.

Drezet, L. E., & Billaut, J. C. (2008). A project scheduling problem with labour constraints and time-dependent activities requirements. *International Journal of Production Economics, 112*, 217 - 225.

Eiben, A. E., & Smith, J. E. (2003). Multimodal problems and spatial distribution. In A. E. Eiben, & J. E. Smith, *Introduction to Evolutionary Computing* (pp. 155-181). Berlin, Germany: Springer-Verlag Berlin Heidelberg.

El Sakkout, H., & Wallace, M. (2000). Probe backtrack search for minimal perturbation in dynamic scheduling. *Constraints, 5*(4), 359–388.

Elazim, S. A., & Ali, E. (2016). Optimal Power System Stabilizers design via Cuckoo Search algorithm. *International Journal of Electrical Power & Energy Systems, 75*(2), 99–107.

Elmaghraby, S. E., Ferreira, A. A., & Tavares, L. (2000). Optimal start times under stochastic activity durations. *International Journal of Production Economics, 64*(1), 153-164.

Escudero, L. F., Kamesam, P. V., King, A. J., & Wets, R. (1993). Production planning via scenarion modelling. *Annals of Operations Research, 43*, 311-335.

Fayek, M. B., Darwish, N. M., & Ali, M. M. (2010). Context based clearing procedure: A niching method for genetic algorithms. *Journal of Advanced Research, 1*, 301–307.

Feller, W. (1968). *An Introduction to Probability Theory and Its Applications.* New-York: Wiley,.

Feo, T. A., & Resende, M. G. (1995). Greedy randomised adaptive search procedures. *Journal of Global Optimisation, 6*, 109-133.

Festa, P., & Resende, M. G. (2002). GRASP: An annotated bibliography. In C. C. Ribeiro, & H. P., *Essays and Survey on Metaheuristics* (pp. 325–367). Bouston, MI: Springer US.

Fleszar, K., & Hindi, K. (2004). Solving the resource-constrained project scheduling problem by a variable neighborhood search. *European Journal of Operational Research, 155*, 402-413.

Fleszar, K., Osman, H. I., & Hindi, K. S. (2009). A variable neighbourhood search algorithm for the open vehicle routing problem. *European Journal of Operational Research, 195*(3), 803–809.

Fonseca, G. H., & Santos, H. G. (2014). Variable Neighborhood Search based algorithms for high school timetabling. *Computers & Operations Research, 52*, 203–208.

Gambardella, L. M., & Dorigo, M. (2000). Ant colony systems hybridized with a new local search for the sequential ordering problems. *Journal of Computing, 12*(3), 2027-2032.

Gan, J., & Warwick, K. (2001). Dynamic Niche Clustering: a fuzzy variable radius niching technique for multimodal optimisation in GAs. *Proceedings of the 2001 Congress on Evolutionary Computation.* Seoul, South Korea: IEEE.

Gao, H. (1995). *Building robust schedules using temporal protection – an empirical study of constraint based scheduling under machine failure uncertainty.* Master's Thesis: Department of Industrial Engineering, University of Toronto.

Gendreau, M., Laporte, G., & Potvin, J.-Y. (2001). Metaheuristics for the vehicle routing problem. *SIAM Series on Discrete Mathematics and Applications, 9*, 129-154.

Ghosh, S. (1996). *Enhancing Real-Time Schedules to Tolerate Transient Faults.* University of Pittsburgh : Ph.D. thesis.

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decimal Science, 8*, 156-166.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research, 13*, 533-549.

Glover, F., & Kochenberger, G. (2005). *Handbook on meteheuristics.* Springer.

Glover, F., Laguna, M., & Marti, R. (2000). Fundamentals of scatter search and path relinking. *Control, 29*(3), 653-684.

Goldberg, D. E. (1987). *Genetic Algorithms in Search, Optimization, and Machine Learning.* New-York: Addison-Wesley.

Goldberg, D. E., & Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimisation. *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application* (pp. 41-49). Cambridge, Massachusetts, USA : L. Erlbaum Associates Inc.

Goldberg, D. E., & Wang, L. (1998). Adaptive niching via coevolutionary sharing. In D. Quagliarella, *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science* (pp. 21-38). New-York: John Wiley and Sons.

Golenko-Ginzburg, D., & Gonik, A. (1997). Stochastic network project scheduling with non-consumable limited resources. *International Journal of Production Economics, 48*, 29-37.

Graham, R. L. (1966). Bounds on multiprocessing timing anomalies. *Bell System Technical Journal, 45*, 1563–1581.

Greenwood, P. E., & Nikulin, M. S. (1996). *A Guide to Chi-Squared Testing.* New York: Wiley.

Grefenstette, J., Gopal, R., Rosmaita, B., & Gucht, D. (1985). Genetic Algorithms for the Traveling Salesman Problem. *Proceedings of the 1st International Conference on Genetic Algorithms* (pp. 160-168). Hillsdale, NJ: L. Erlbaum Associates Inc.

Haitao, L., & Womer, N. K. (2015). Solving stochastic resource-constrained project scheduling problems by closed-loop approximate dynamic programming. *European Journal of Operational Research, 246*, 20-33.

Hansen, P., & Mladenovic, N. (1999). An introduction to variable neighborhood search. In *Metaheuristics: Advances and trends in local search paradigms for optimization* (pp. 433–458). Kluwer Academic Publishers.

Hapke, M., & Slowinski, R. (1996). Fuzzy priority heuristics for project scheduling. *Fuzzy Sets and Systems, 83*, 291-299.

Harik, G. R. (1997). Finding multi-modal solutions using restricted tournament selection. *Proceedings of the Sixth International Conference on Genetic Algorithms* (pp. 24-31). San Francisco, CA: Morgan Kaufmann Publishers Inc.

Hartmann, S. (1998). A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics , 45*(7), 733–750.

Hartmann, S. (2002). A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics , 49*(5), 433–448.

Hartmann, S., & Briskorn, D. (2010). A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research, 207*, 1–14.

Hartmann, S., & Kolisch, R. (2000). Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling. *European Journal of Operational Research, 127*(2), 394–407.

Herroelen, W. (2006). Project scheduling–theory and practice. *Production and Operations Management, 14*(4), 413–432.

Herroelen, W., & Leus, R. (2004). The construction of stable project baseline schedules. *European Journal of Operational Research, 156*, 550–565.

Herroelen, W., & Leus, R. (2005). Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research, 165*, 289-306.

Hindi, K. S., Yang, H., & Fleszar, K. (2002). An evolutionary algorithm for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation, 6*, 512-518.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* Michigan: MIT Press.

Holland, J. H. (1975). Interim and prospectus. In *Adaptation in Natural and Artificial Systems* (pp. 171–180). Ann Arbor, MI: Univ. of Michigan Press.

Horn, J. (2002). Resource-Based Fitness Sharing. In *Volume 2439 of the series Lecture Notes in Computer Science* (pp. 381-390). Berlin : Springer Berlin Heidelberg.

Husbands, P., Jermy, G., McIlhagga, M., & Ives, R. (1996). Two applications of genetic algorithms to component design. *AISB workshop on evolutionary computing*, (pp. 50-61). Chicago, CH.

Igelmund, G., & Radermacher, F. J. (1983). Preselective strategies for the optimization of stochastic project networks under resource constraints. *Networks, 13*, 1-28.

Ikeda, K., & Kobayashi, S. (2000). GA Based on the UV-Structure Hypothesis and Its Application to JSP. In *Parallel Problem Solving from Nature PPSN VI* (pp. 273-282). Berlin: Springer Berlin Heidelberg.

Iwamatsu, M. (2006). Multi-species particle swarm optimizer for multimodal function optimization. *IEICE Transactions on Information and Systems, E89D*(3), 1181–1188.

Jelasity, M., Ortigosa, P. M., & García, I. (2001). UEGO, an Abstract Clustering Technique for Multimodal Global Optimization. *Journal of Heuristics, 7*(3), 215-233.

Jones, T. (1995). *Evolutionary Algorithms, Fitness Landscapes and Search.* Albuquerque, New Mexico: PhD thesis, University of New Mexico,.

Jong, K. D. (1975). *An analysis of the behavior of a class of genetic adaptive systems.* Ann Arbor, MI: Ph.D. Dissertation, Univ. Michigan.

Jozefowksa, J., Mika, M., Rozycki, R., Waligora, G., & Weglarz, J. (2001). Simulated Annealing for Multi-Mode Resource-Constrained Project Scheduling. *Annals of Operations Research, 102*, 137–155.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks* (pp. 1942-1948). Piscataway, NJ: IEEE.

Kilby, P., Prosser, P., & Shaw, P. (1999). Guided Local Search for the Vehicle Routing Problem with time windows. In *Meta-heuristics: Advances and trends in local search paradigms for optimization* (pp. 473–486). Berlin: Eds. Kluwer Academic.

Kirkpatrick, S., Gelatt, C., & Vecchi, M. P. (1983). Optimisation by simulated annealing. *Science, 13*, 671-680.

Klein, R. (2000). Project scheduling with time-varying resource constraints. *International Journal of Production Research, 38*(16), 3937–3952.

Klein, R. (2001). *Scheduling of resource-constrained projects.* Kluwer Academic Publishers.

Kochetov, Y. A., & S. (2011). Iterative local search methods for the talent scheduling problem. *Proceedings of 1st international symposium and 10th Balkan conference on operational research, September 22*, (pp. 282-288). Thessaloniki, Greece .

Kochetov, Y. A., & Stolyar, A. A. (2003). Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem. *Proceedings of the 3rd International Workshop of Computer Science and Information Technologies.* Novosibirks, Russia.

Kolisch, R. (1996). Serial and parallel resource-constrained project scheduling: theory and computation. *European Journal of Operational Research, 90*(2), 320–333.

Kolisch, R., & Hartmann, S. (1999). Heuristic Algorithms for the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis. In *Project Scheduling: Recent models, algorithms and applications* (pp. 147-178). Berlin: Kluwer.

Kolisch, R., & Hartmann, S. (2006). Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research, 174*(1), 23–37.

Kolisch, R., & Padman, R. (2001). An integrated survey of deterministic project scheduling. *Omega, 29*, 249-272.

Kolisch, R., & Sprecher, A. (1997). PSPLIB - A project scheduling problem library. *European Journal of Operational Research, 96*(1), 205–216.

Kolisch, R., Sprecher, A., & Drexl, A. (1995). Characterization and generation of a general class of resource-constrained project scheduling problems. *Management Science, 41*(10), 1693-1703 .

Kratica, J., Tosic, D., Filipovic, V., & Dugosija, D. (2011). A new genetic representation for quadratic assignment problem. *Yugoslav Journal of Operations Research, 21*(2), 225-238.

Kundu, S., Biswas, S., Das, S., & Suganthan, P. N. (2013). Crowding-based local Differential Evolution with Speciation-based Memory Archive for Dynamic Multimodal Optimization. *Proceedings of the 15th annual conference on Genetic and evolutionary computation* (pp. 33-40). New York, NY: ACM.

Laarhoven, P. J., Aarts, E. H., & Lenstra, J. K. (1992). Job Shop Scheduling by Simulated Annealing. *Operations Research, 40*, 113-125.

Laguna, M., Lourenco, H., & Marti, R. (2000). Assigning proctors to exams with Scatter Search. In *Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research* (pp. 215–227). Boston: Eds. Kluwer Academic Publishers, MA.

Lamas, P., & Demeulemeester, E. (2015). A purely proactive scheduling procedure for the resource-constrained project scheduling problem with stochastic activity durations. *Journal of Scheduling, 193*, 1-20.

Lawrence, S. R. (1984). *Resource-Constrained Project Scheduling: an experimental investigation of heuristic scheduling techniques.* Graduate School of Industrial Administration. Pittsburgh, PA, USA: Carnegie-Mellon University.

Leccardi, M. (2005). Comparison of three algorithms for Lèvy Noise Generation. *Proceedings of fifth EUROMECH nonlinear dynamics conference* (pp. 5-11). Eindhoven.

Leeuwen, J. v. (1998). Handbook of Theoretical Computer Science. In *Vol. A, Algorithms and complexity.* Amsterdam: Elsevier.

Li, C.-S., Priemer, R., & Cheng, K.-H. (2004). Optimization by random search with jumps. *International Journal for Numerical Methods in Engineering, 60*(7), 1301-1315.

Li, J., & Wood, A. (2009). Random search with species conservation for multimodal functions. *IEEE Congress on Evolutionary Computation.* Trondheim, Norway: IEEE. doi:10.1109/CEC.2009.4983344

Li, J.-P., Balazs, M. E., & Parks, G. T. (2002). A species conserving genetic algorithm for multimodal function optimization. *Evolutionary Computation, 10*(3), 207-234.

Li, X.-D. (2004). Adaptively Choosing Neighbourhood Bests using Species in a Particle Swarm Optimizer for Multimodal Function Optimisation. *Lecture Notes in Computer Science, 3102*, 105–116.

Linyi, D., & Lin, Y. (2007). A Particle Swarm Optimization for Resource-Constrained Multi-Project Scheduling Problem. *International Conference on Computational Intelligence and Security* (pp. 1010-1014). Harbin, China: IEEE.

Lourenco, H. R., Martin, O., & Stutzle, T. (2001). A beginner's introduction to Iterated Local Search. *Proceeding of the 4th Metaheuristics International Conference*, (pp. 1-11). Porto, Portugal.

Luo, S., Wang, C., & Wang, J. (2003). Ant colony optimization for resource-constrained project scheduling with generalized precedence relations. *International Conference on Tools with Artificial Intelligence.* Sacramento, CA, USA: IEEE. doi:10.1109/TAI.2003.1250202

MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations. *University of California Press.* University of California.

Mahdi-Mobini, M. D., Rabbani, M., Amalnik, M. S., Razmi, J., & Rahimi-Vahed, A. R. (2009). Using an enhanced scatter search algorithm for a resource-constrained project scheduling problem. *Soft Computing, 13*(6), 597-610.

Mahfoud, S. W. (1992). Crowding and preselection revisited. *Parallel Problem Solving from Nature, 2*, pp. 27-37.

Mahfoud, S. W. (1995). A Comparison of Parallel and Sequential Niching Methods. *Proceedings of the 6th International Conference on Genetic Algorithms* (pp. 136-143). San Francisco, CA: Morgan Kaufmann Publishers Inc,.

Malcolm, D. G., Roseboom, J. H., Clark, C. E., & Fazar, W. (1959). Applications of a technique for research and development program evaluation. *OperationsResearch, 7*(5), 646-669.

Mantegna, R. N. (1994). Fast, accurate algorithm for numerical simulation of Levy stable stochastic processes. *Physical Review E, 49*, 4677-4683.

Marti, R., Laguna, M., & Campos, V. (2005). Scatter search vs. genetic algorithms. An experimental evaluation with permutation problems. In *Metaheuristic Optimization Via Memory and Evolution. Operations Research/Computer Science Interfaces Series.* (pp. 263–282). Boston, MA: Kluwer Academic Publishers.

Martin, O., Otto, S., & Felten, E. W. (1991). Large-step Markov chains for the traveling salesman problem. *Complex Systems, 5*(3), 299-326.

Martinez, L., & Soares, S. (2002). Comparison between closed-loop and partial open-loop feedback control policies in long term hydrothermal scheduling. *IEEE Transactions on Power Systems, 17*(2), 330 - 336.

Mehta, S., & Uzsoy, R. (1998). Predictable Scheduling of a Job Shop Subject to Breakdowns. *IEEE Transactions on Robotics and Automation, 14*(3), 365-378.

Mendes, J. J., Gonçalves, J. F., & Resende, M. G. (2009). A random key based genetic algorithm for the resource constrained project scheduling problem. *Computers and Operations Research, 36*(1), 92-109.

Merkle, D., Middendorf, M., & Schmeck, H. (2002). Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation, 6*(4), 333-346.

Mills, P., & Tsang, E. (2000). Guided local search for solving SAT and weighted MAX-SAT Problems. *Journal of Automated Reasoning, 24*(1), 205–223.

Mingozzi, A., Maniezzo, V., Ricciardelli, S., & Bianco, L. (1998). An Exact Algorithm for the Resource Constrained Project Scheduling Problem Based on a New Mathematical Formulation. *Management Science, 44*(5), 714-729 .

Mohring, R. H., & Stork, F. (2000). Linear preselective policies for stochastic project scheduling. *Mathematical Methods of Operations Research, 52*(3), 501–515.

Mohring, R. H., Radermacher, F., & Weiss, G. (1984). Stochastic scheduling problems I – General strategies. *Operations Research, 28*, 65-104.

Mori, M., & Tseng, C. C. (1997). A genetic algorithm for multi-mode resource constrained project scheduling problem. *European Journal of Operational Research, 100*(1), 134–141.

Moscato, P. (1989). *On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms.* Report 826: CalTech Concurrent Computation Program, Tech. Rep.

Moumene, K., & Ferland, J. A. (2008). New representation to reduce the search space for the resource-constrained project scheduling problem. *RAIRO - Operations Research, 42*(2), 215-228.

Neumann, K., Schwindt, C., & Zimmermann, J. (2003). *Project Scheduling with Time Windows and Scarce Resources.* Berlin: Springer-Verlag.

Nguyen, T. T., Truong, A. V., & Phung, T. A. (2016). A novel method based on adaptive cuckoo search for optimal network reconfiguration and distributed generation allocation in distribution network. *International Journal of Electrical Power & Energy Systems, 78*(6), 801–815.

Nikulin, Y., & Drexl, A. (2010). Theoretical aspects of multicriteria flight gate scheduling: deterministic and fuzzy models. *Journal of Scheduling, 13*, 261-281.

Nonobe, K., & Ibaraki, T. (2002). Formulation and tabu search algorithm for the resource constrained project scheduling problem. In C. C. Hansen, *Essays and surveys in metaheuristics* (pp. 557–588). Springer US.

Nowicki, E., & Smutnicki, C. (1996). A fast taboo search algorithm for the job-shop problem. *Management Science, 42*(2), 797-813.

Olagu´ibel, R. A.-V., & Goerlich, J. M. (1993). The project scheduling polyhedron: Dimension, facets, and lifting theorems. *European Journal of Operational Research, 67*, 204–220.

Ouaarab, A., Ahiod, B., & Yang, X.-S. (2013). Discrete cuckoo search algorithm for the travelling salesman problem. *Neural Computing and Applications, 24*(7), 1659-1669.

Ozdamar, L., & Ulusoy, G. (1995). A survey on the resource–constrained project scheduling. *IIE Transactions, 27*, 574–586.

Palmer, C., & Kershenbaum, A. (1994). Representing trees in genetic algorithms. *Proceedings of the first IEEE international conference on evolutionary computation* (pp. 379-384). New York: IEEE Press.

Palpant, M., Artigues, C., & Michelon, P. (2004). LSSPER: Solving the Resource-Constrained Project Scheduling Problem with Large Neighbourhood Search. *Annals of Operations Research, 131*(1), 237-257.

Pan, N.-H., Lee, M., & Chen, K.-Y. (2009). Improved Tabu Search Algorithm Application in RCPSP. *Proceedings of the International MultiConference of Engineers and Computer Scientists,. Vol I*, p. 44. Hong-Kong: IMECS 2009, March 18-20.

Paraskevopoulos, D. C., Tarantilis, C. D., & Ioannou, G. (2012). Solving project scheduling problems with resource constraints via an event list-based evolutionary algorithm. *Expert Systems with Applications, 39*, 3983-3994.

Parrot, D., & Li, X. (2004). A particle swarm model for tracking multiple peaks in a dynamic environment using speciation. *Congress on Evolutionary Computation, 19-23 June 2004.* Portland, OR, USA: IEEE. doi:10.1109/CEC.2004.1330843

Passaro, A., & Starita, A. (2008). Particle swarm optimization for multimodal functions: clustering approach. *Journal of Artificial Evolution and Applications, 2008*(8).

Pavlyukevich, I. (2007). Lévy flights for Non-local search and simulated annealing. *Journal of Computational Physics*, 1830-1844.

Payne, R. B., & Sorenson, M. D. (2005). *The Cuckoos.* Oxford, UK: Oxford University Press.

Pérez, A., Posada, M., & Lorenzana, A. (2015). Taking advantage of solving the resource constrained multi-project scheduling problems using multi-modal genetic algorithms. *Soft Computing, 20*(5), 1879–1896.

Pérez, E., Herrera, F., & Hernández, C. (2003). Finding multiple solutions in job shop scheduling by niching genetic algorithms. *Journal of Intelligent Manufacturing, 14*(3), 323–339.

Pérez, E., Posada, M., & Herrera, F. (2012). Analysis of new niching genetic algorithms for finding multiple solutions in the job shop scheduling. *Journal of Intelligent Manufacturing, 23*(3), 341–356.

Pesek, I., Schaerf, A., & Zerovnik, J. (2007). Hybrid local search techniques for the resource-constrained project scheduling problem. *LNCS, 4771*, 57-68.

Petrowski, A. (1996). A clearing procedure as a niching method for genetic algorithms. *Proceedings of Third IEEE International Conference on Evolutionary Computation.* Nagoya, Japan: IEEE. doi:10.1109/CEC.1996.542703

Prais, M., & Ribeiro, C. C. (2000). Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *Journal of Computing, 12*, 164-176.

Qing, L., Gang, W., Zaiyue, Y., & Qiuping, W. (2008). Crowding clustering genetic algorithm for multimodal function optimisation. *Applied Soft Computing, 8*, 88-95.

Qu, B.-Y., & Suganthan, P. N. (2010). Novel multimodal problems and differential evolution with ensemble of restricted tournament selection. *IEEE Congress on Evolutionary Computation, 18-23 July 2010.* Barcelona, Spain: IEEE. doi:10.1109/CEC.2010.5586341

Qu, B.-Y., Liang, J., Suganthan, P., & Chen, T. (2012). Ensemble of clearing differential evolution for multi-modal optimization. In Y. Tan, Y. Shi, & Z. (. Ji, *Advances in Swarm Intelligence. ICSI 2012. Lecture Notes in Computer Science* (Vol. vol. 7331, pp. 350-357). Berlin: Springer Berlin Heidelberg.

Radermacher, F. J. (1981). Cost-dependent essential systems of ES-strategies for stochastic scheduling problems. *Methods of Operations Research, 42*, 17–31.

Radermacher, F. J. (1984). Optimal strategies for stochastic scheduling problem. A survey. In S. Boroda, *Writings on computer science and applied mathematics* (pp. 114-201). RWTH Aachen: Berlin.

Radermacher, F. J. (1985). Scheduling of project networks. *Annals of Operations Research, 4*, 227-252.

Ranjbar, M. (2008). Solving the resource constrained project scheduling problem using filter-and-fan approach. *Applied Mathematics and Computation, 201*, 313-318.

Ranjbar, M., & Kianfar, F. (2009). A hybrid scatter search for the RCPSP. *Transaction E: Industrial Engineering, 16*(1), 11-18.

Reeves, C. R. (1999). Landscapes, operators and heuristic search. *Annals of Operations Research, 86*(1), 473–490.

Resende, M. G., & Ribeiro, C. C. (2001). A GRASP for graph planarisation. *Networks, 27*(3), 201-222.

Rockafellar, R. T., & Wets, R. J.-B. (1991). Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research, 16*, 119-147.

Ronald, S. (1998). More distance functions for order-based encodings. *IEEE World Congress on Computational Intelligence.* Anchorage, AK, USA: IEEE. doi:10.1109/ICEC.1998.700089

Roshanaei, V., Naderi, B., Jolai, F., & Khalili, M. (2009). A variable neighborhood search for job shop scheduling with set-up times to minimize makespan. *Future Generation Computer Systems, 25*(6), 654–661.

Roy, R., & Parmee, I. C. (1996). Adaptive restricted tournament selection for the identification of multiple sub-optima in a multi-modal function. In

Lecture Notes in Computer Science, vol. 1143 (pp. 236–256). London: Springer-Verlag.

Sacco, W. F., Henderson, N., & Rios-Coelho, A. C. (2014). Topographical clearing differential evolution: A new method to solve multimodal optimization problems. *Progress in Nuclear Energy, 71*, 269-278.

Sacco, W. F., Machado, M. D., Pereira, C. M., & Schirru, R. (2004). The fuzzy clearing approach for a niching genetic algorithm applied to a nuclear reactor core design optimization problem. *Annals of Nuclear Energy, 31*, 55-69.

Sadeh-Koniecpol, N., & Otsuka, S. (1993). Predictive and reactive scheduling with the microboss production scheduling and control system. *Proceedings of the IJCAI-93 Workshop on Knowledge-based Production Planning, Scheduling, and Control.* Chambery, France.

Salazar-Lechuga, M., & Rowe, J. E. (2005). Particle swarm optimization and fitness sharing to solve multi-objective optimization problems. *IEEE Congress on Evolutionary Computation , 2*, 1204 - 1211.

Sekhar, P., & Mohanty, S. (2016). An enhanced cuckoo search algorithm based contingency constrained economic load dispatch for security enhancement. *International Journal of Electrical Power & Energy Systems, 75*(2), 303–310.

Sevaux, M., & Sorensen, K. (2002). A genetic algorithm for robust schedules in a just-in-time environment. *8th International Workshop on Project Management and Scheduling*, (pp. 16-32). Valencia.

Shahsavar, M., & AkhavanNiaki, S. T. (2010). An efficient genetic algorithm to maximize net present value of project payments under inflation and bonus–penalty policy in resource investment problem. *Advances in Engineering Software, 41*, 1023–1030.

Sheng, W., Liu, X., & Fairhurst, M. (2008). A Niching Memetic Algorithm for Simultaneous Clustering and Feature Selection. *IEEE Transactions on Knowledge and Data Engineering, 20*(7), 868 - 879.

Shi, X., Liang, Y., Lee, H., Lu, C., & Wang, Q. (2007). Particle swarm optimization-based algorithms for TSP and generalized TSP. *Information Processing Letters, 105*(3), 169–176.

Shibasaka, M., Hara, A., Ichimura, T., & Takahama, T. (2007). Species-based differential evolution with switching search strategies for multimodal function optimization. *IEEE Congress on Evolutionary Computation, 25-28 Sept 2007.* Singapore: IEEE. doi:10.1109/CEC.2007.4424604

Shlesinger, M. F., Zaslavsky, G. M., & Frisch, U. (1994). Lévy Flights and Related Topics in Physics. *Proceedings of the International Workshop. Lecture Notes in Physics, Vol. 450.* Nice, France: Springer-Verlag Berlin Heidelberg.

Slowinski, R., & Hapke, M. (2010). *Scheduling under Fuzziness.* Physica-Verlag: Heidelberg.

Smith, S. F. (1994). Reactive scheduling systems. In D. Brown, & W. Scherer, *Intelligent Scheduling Systems* (pp. 155-169). Kluwer US.

Sprecher, A. (2000). Scheduling Resource-Constrained Projects Competitively at Modest Memory Requirements. *Management Science, 46*(5), 710-723.

Sprecher, A., Kolisch, R., & Drexl, A. (1995). Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem. *European Journal of Operational Research, 80*(1), 94–102.

Stoean, C., Preuss, M., Stoean, R., & Dumitrescu, D. (2010). Multimodal Optimization by Means of a Topological Species Conservation Algorithm. *IEEE Transactions on Evolutionary Computation, 14*(6), 842 - 864.

Stork, F. (2001). *Stochastic resource-constrained project scheduling.* Technische Universitat Berlin.: Ph.D. thesis.

Streichert, F., Stein, G., Ulmer, H., & Zell, A. (2004). A Clustering Based Niching EA for Multimodal Search Spaces. In *Artificial Evolution* (pp. 293-304). Berlin : Springer Berlin Heidelberg.

Stutzle, T. G. (1999). *Local Search Algorithms for Combinatorial Problems - Analysis, Improvements and New Applications* (Vol. 220 of Dissertations in Artificial Intelligence). IOS Press, Incorporated.

Surekha, P., Raajan, P. M., & Sumathi, S. (2010). Particle Swarm Optimization approaches to solve combinatorial job shop scheduling problems. *IEEE International Conference on Computational Intelligence and Computing Research* (pp. 1-5). Coimbatore, India: IEEE.

Taillard, E. (1991). Robust Taboo Search for the Quadratic Assignment Problem. *Parallel Computations, 17*, 443–455.

Talbi, E.-G. (2002). A Taxonomy of Hybrid Metaheuristics. *Journal of Heuristics, 8*(5), 5410564.

Talbot, F. B. (1982). Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case. *Management Science, 28*, 1197-1210.

Teymourian, E., V.Kayvanfar, Komaki, G., & Zadeha, M. (2016). Enhanced intelligent water drops and cuckoo search algorithms for solving the capacitated vehicle routing problem. *Information Sciences, 334*(6), 354–378.

Thomas, P. R., & Salhi, S. (1998). A tabu search approach for the resource constrained project scheduling problem. *Journal of Heuristics, 4*, 123-139.

Thomsen, R. (2004). Multimodal optimization using crowding-based differential evolution. *Proceedings of the Congress on Evolutionary Computation.* Portland, OR, USA: IEEE. doi:10.1109/CEC.2004.1331058

Tolku, Y. C. (2002). Application of genetic algorithms to construction scheduling with or without resource constraints. *Canadian Journal of Civil Engineering, 29*, 421-429.

Tormos, P., & Lova, A. (2001). A Competitive Heuristic Solution Technique for Resource-Constrained Project Scheduling. *Annals of Operations Research, 102*(1), 65-81.

Tsai, Y.-W., & Gemmill, D. D. (1998). Using tabu search to schedule activities of stochastic resource-constrained projects. *European Journal of Operational Research, 111*, 129-141.

Tseng, L., & Chen, S. (2006). A hybrid metaheuristic for the resource-constrained project scheduling problem. *European Journal of Operational Research, 175*(2), 707-721.

Valls, V., Ballestin, F., & Quintanilla, S. (2003). *A hybrid genetic algorithm for the RCPSP.* Technical Report, University of Valencia, Department of Statistics and Operations Research.

Valls, V., Ballestín, F., & Quintanilla, S. (2005). Justification and RCPSP: A technique that pays. *European Journal of Operational Research, 165*(2), 375–386.

Valls, V., Ballesting, F., & Quintanilla, S. (2004). A population-based approach to the resource-constrained project scheduling problem. *Annals of Operations Research, 305-324*, 131.

Valls, V., Quintanilla, M. S., & Ballestin, F. (2003). Resource-constrained project scheduling: A critical reordering heuristic. *European Journal of Operational Research, 149*, 282-301.

Viswanathan, G. M. (2008). Lévy flights and superdiffusion in the context of biological encounters and random searches. *Physics of Life Reviews*, 133-150.

Vitela, J. E., & Castanos, O. (2008). A real-coded niching memetic algorithm for continuous multimodal function optimization. *Proceedings of Evolutionary Computation, IEEE World Congress on Computational Intelligence* (pp. 50-62). Hong Kong, China: IEEE.

Vonder, S. V., Demeulemeester, E., & Herroelen, W. (2007). A classification of predictive-reactive project scheduling procedures. *Journal of Scheduling, 10*, 195-207.

Voudoris, C., & Tsang, E. (1999). Guided local search. *European Journal of Operational Research, 113*(2), 469–499.

Weglarz, J. (1999). *Project Scheduling. Recent models, Algorithms and Applications.* Berlin: Kluwer Academic Publishers.

Wets, R. J.-B. (1989). The aggregation principle in scenario analysis and stochastic optimization. In S. W. Wallace, *Algorithms and Model Formulations in Mathematical Programming* (pp. 91-113). Berlin, Heidelberg: Springer.

Wolpert, D., & Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation, 1*(1), 67 - 82. doi:10.1109/4235.585893

Wu, S., Wan, H., Shukla, S. K., & Li, B. (2011). Chaos-based improved immune algorithm (CBIIA) for resource-constrained project scheduling problems. *Expert Systems with Applications, 38*, 3387–3395.

Xiong, J., Leusb, R., Yanga, Z., & Abbass, H. A. (2016). Evolutionary multi-objective resource allocation and scheduling in the Chinese navigation satellite system project. *European Journal of Operational Research, 251*(2), 662–675.

Yang, H. Z., Li, F. C., & Wang, C. M. (2005). A density clustering-based niching genetic algorithm for multimodal optimization. *Proceedings of 2005 International Conference on Machine Learning and Cybernetics, 3*, 1599–1604.

Yang, X. (2010). *Engineering Optimization: An Introduction with Metaheuristic Applications.* Cambridge: John Wiley & Sons.

Yang, X.-S. (2012). Flower Pollination Algorithm for Global Optimisation. *Unconventional Computation and Natural Computation, 7445*, 240-249.

Yang, X.-S., & Deb, S. (2009). Cuckoo search via Levy Flights. *Proc. of World congress on Nature & Biologically Inspired Computing* (pp. 210-214). Coimbatore, India: IEEE.

Yang, X.-S., & Deb, S. (2010). Engineering Optimisation by Cuckoo Search. *Mathematical Modelling and Numerical Optimisation, 1*(4), 330-343.

Yilmaz, A. E., & Kuzuoglu, M. (2009). A particle swarm optimization approach for hexahedral mesh smoothing. *International Journal for Numerical Methods in Fluids, 60*, 55–78.

Yin, X., & Germay, N. (1993). A fast genetic algorithm with sharing scheme using cluster analysis methods in multi-modal function optimization. In R. Albrecht, C. Reeves, & N. Steele, *Artificial Neural Nets and Genetic Algorithms.* Vienna, Austria: Springer.

Yuan, Y., Wang, K., & Ding, L. (2009). A Solution to Resource-Constrained Project Scheduling Problem: Based on Ant Colony Optimization Algorithm. *Ninth International Conference on Hybrid Intelligent Systems.* Shenyang, China: IEEE. doi:10.1109/HIS.2009.91

Zaharie, D. (2004). Extensions of differential evolution algorithms for multimodal optimization. *Proceedings of SYNASC'04, 6th International Symposium of Symbolic and Numeric Algorithms for Scientific Computing*, (pp. 523-534).

Zhang, H. (2005). Particle swarm optimization-based schemes for resource-constrained project scheduling. *Automation in Construction, 14*(3), 393–404.

Zhou, Y., & Chen, Y. (2002). Business process assignment optimization. *IEEE International Conference on Systems, Man and Cybernetics.* Yasmine Hammamet, Tunisia: IEEE.

Zhu, J., Li, X., & Shen, W. (2011). Effective genetic algorithm for resource-constrained project scheduling with limited preemptions. *International Journal of Machine Learning and Cybernetics*(2), 55-65.